

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-097205

(43)Date of publication of application : 08.04.1997

(51)Int.Cl.

G06F 12/00

G06F 12/00

G11C 16/06

(21)Application number : 07-251240

(71)Applicant : CANON INC

(22)Date of filing : 28.09.1995

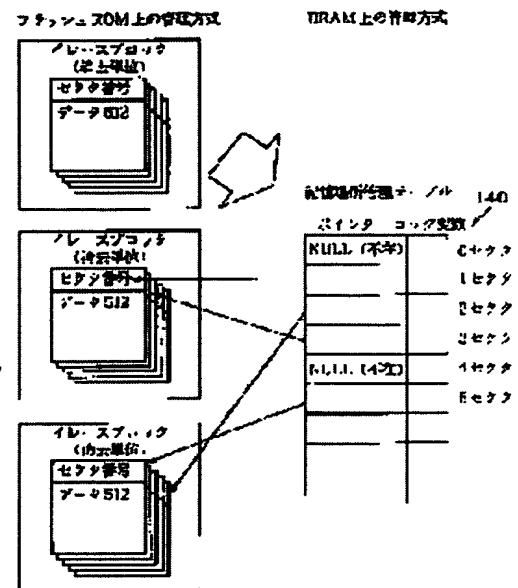
(72)Inventor : OGAWA TAKESHI

(54) METHOD, DEVICE FOR MANAGING FLASH ROM AND COMPUTER CONTROL EQUIPMENT

(57)Abstract:

PROBLEM TO BE SOLVED: To let a flash ROM suit a filling system by inhibiting access from a task to a storage block when that block is accessed by the other task.

SOLUTION: The reading of an Nth sector is started and the Nth sector is locked by using a lock parameter. This lock parameter is managed by a storage place managing table 140. When any sector is already locked by the other sector, this Nth sector is locked after that sector was unlocked by the other task. The locked sector can be occupied by the present tank until it is unlocked. When a logical sector is locked, it is confirmed, while referring to the storage place managing table, whether effective data are stored in this sector or not. When the data have not been stored, a dummy data are read out as the contents of the sector. When the data have been stored, on the other hand, the data are read out of the flash ROM (or a main memory) based on the value of the recording place managing table.



LEGAL STATUS

[Date of request for examination]

27.09.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-97205

(43) 公開日 平成9年(1997)4月8日

(51) Int. Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所	
G06F 12/00	572		G06F 12/00	572	A
	535			535	Z
G11C 16/06			G11C 17/00	309	F

審査請求 未請求 請求項の数10 O L (全43頁)

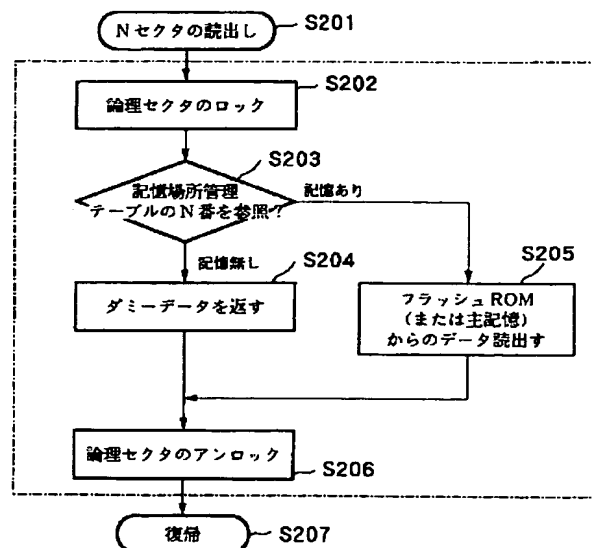
(21) 出願番号	特願平7-251240	(71) 出願人	000001007 キヤノン株式会社 東京都大田区下丸子3丁目30番2号
(22) 出願日	平成7年(1995)9月28日	(72) 発明者	小川 武志 東京都大田区下丸子3丁目30番2号 キヤ ノン株式会社内
		(74) 代理人	弁理士 大塚 康德 (外1名)

(54) 【発明の名称】 フラッシュROM管理方法及び装置及びコンピュータ制御装置

(57) 【要約】

【課題】 フラッシュROMをファイルシステムに適応させることが可能な管理方式を、マルチタスクシステムに好適に適用することを可能とする。

【解決手段】 データ領域とこれに対応する管理領域とでセクタを構成し、複数のセクタをフラッシュROMに形成し、各セクタにおいてデータ領域の記憶状態を示す状態情報(セクタ番号、使用中フラグ、使用済フラグ)を管理領域に格納し、該状態情報に基づいてフラッシュROMのアクセスを管理する。このようなフラッシュROMへのアクセスにおいて、並列に実行可能な複数のタスクの一つによって前記複数のセクタの一つがアクセスされた場合、そのアクセスされたセクタについて、他のタスクからのアクセスを禁止する。



【特許請求の範囲】

【請求項 1】 複数のタスクが並列的に実行されるシステムにおけるフラッシュ ROM の管理装置であって、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理手段と、前記複数のタスクの一つによって前記複数の記憶ブロックの一つがアクセスされた場合、そのアクセスされた記憶ブロックについて、他のタスクからのアクセスを禁止する排他手段とを備えることを特徴とするフラッシュ ROM 管理装置。

【請求項 2】 前記管理手段は、フラッシュ ROM へのアクセスがデータの書込みであった場合、未使用状態の記憶ブロックを検索して、検索された記憶ブロックヘデータの書込みを行い、

前記排他手段は、前記未使用状態の記憶ブロックの検索中は、他のタスクから未使用状態の記憶ブロックの検索を行うことを禁止することを特徴とする請求項 1 に記載のフラッシュ ROM 管理装置。

【請求項 3】 前記管理手段は、前記複数のタスクによる複数のフラッシュ ROM への書込みをラウンドロビンスケジューリングによる時分割多重で同時に行うことを特徴とする請求項 1 に記載のフラッシュ ROM 管理装置。

【請求項 4】 複数のタスクを優先度を持たせて並列的に実行させることが可能なシステムにおけるフラッシュ ROM の管理装置であって、

データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理手段と、状態情報が無効データを示す記憶ブロックに対して消去速度を向上するための前処理を行う前処理手段と、前記前処理手段の実行を前記複数のタスクのうちの低優先度のタスクに割り当てて実行する実行手段とを備えることを特徴とするフラッシュ ROM 管理装置。

【請求項 5】 複数のタスクが並列的に実行されるシステムにおけるフラッシュ ROM の管理方法であって、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理工程と、前記複数のタスクの一つによって前記複数の記憶ブロックの一つがアクセスされた場合、そのアクセスされた記憶ブロックについて、他のタスクからのアクセスを禁止する排他工程とを備えることを特徴とするフラッシュ ROM 管理方法。

OM 管理方法。

【請求項 6】 前記管理工程は、フラッシュ ROM へのアクセスがデータの書込みであった場合、未使用状態の記憶ブロックを検索して、検索された記憶ブロックヘデータの書込みを行い、

前記排他工程は、前記未使用状態の記憶ブロックの検索中は、他のタスクから未使用状態の記憶ブロックの検索を行うことを禁止することを特徴とする請求項 5 に記載のフラッシュ ROM 管理方法。

【請求項 7】 前記管理工程は、前記複数のタスクによる複数のフラッシュ ROM への書込みをラウンドロビンスケジューリングによる時分割多重で同時に行うことを特徴とする請求項 5 に記載のフラッシュ ROM 管理方法。

【請求項 8】 複数のタスクを優先度を持たせて並列的に実行させることが可能なシステムにおけるフラッシュ ROM の管理方法であって、

データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理工程と、状態情報が無効データを示す記憶ブロックに対して消去速度を向上するための前処理を行う前処理工程と、前記前処理工程の実行を前記複数のタスクのうちの低優先度のタスクに割り当てて実行する実行工程とを備えることを特徴とするフラッシュ ROM 管理方法。

【請求項 9】 メモリ媒体から所定のプログラムを読み込んでコンピュータを制御するコンピュータ制御装置であって、前記メモリ媒体は、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理工程の手順コードと、

並列に実行可能な複数のタスクの一つによって前記複数の記憶ブロックの一つがアクセスされた場合、そのアクセスされた記憶ブロックについて、他のタスクからのアクセスを禁止する排他工程の手順コードとを備えることを特徴とするコンピュータ制御装置。

【請求項 10】 メモリ媒体から所定のプログラムを読み込んでコンピュータを制御するコンピュータ制御装置であって、前記メモリ媒体は、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュ ROM に形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュ ROM のアクセスを管理する管理工程の手順コードと、

状態情報が無効データを示す記憶ブロックに対して消去速度を向上するための前処理を行う前処理工程の手順コードと、

付与された優先度に従って実行がスケジューリングされる複数のタスクのうちのタスクのうちの低優先度のタスクに前記前処理工程の実行を割り当てて実行する実行工程の手順コードとを備えることを特徴とするコンピュータ制御装置。

【発明の詳細な説明】

【 0 0 0 1 】

【発明の属する技術分野】本発明は、コンピュータ等におけるフラッシュROMの管理方法及び装置及びコンピュータ制御装置に関する。

【 0 0 0 2 】

【従来の技術】フラッシュROMは現在いろいろなタイプのものがあるが大きく分けてフラッシュDISK用に開発されたタイプとパーソナルコンピュータのBIOS用に開発されたものがある。

【 0 0 0 3 】前者は消去単位がハードディスクで一般的な512バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュROMは消去単位が例えば64Kなどといった大きなブロック単位でしか行えない様になっている。また、PROMの様に書き込み電圧として12V等の電圧が必要なものもある。後者のフラッシュROMの方が安価に入手できるがファイルシステムとの整合性が悪い為に特に小容量の記録メディアとしては使う事が出来なかった。

【 0 0 0 4 】

【発明が解決しようとする課題】以上のように、BIOS用に設計されたフラッシュROMは、その消去単位が大きく、ファイルシステムとの整合性が悪いが、安価で入手しやすい。従って、そのようなフラッシュROMをファイルシステムに適用できれば、安価な小容量の記録メディアを提供することができる。

【 0 0 0 5 】本発明は、消去単位の大きいフラッシュROMをファイルシステムと適合させることを可能とするフラッシュROM管理方法及び装置を提供しようとするものである。

【 0 0 0 6 】フラッシュROMへのデータの書き込み、消去等は、CPUの動作速度に対して非常に時間のかかる処理である。このため、複数のタスクを並列的に実行するマルチタスク処理において、フラッシュROMへのアクセス管理を行う場合、何等かの対処が必要となる。

【 0 0 0 7 】本発明は上記の問題に鑑みてなされたものであり、フラッシュROMをファイルシステムに適合させることが可能な管理方式を、マルチタスクシステムに好適に適用することを可能とするフラッシュROM管理方法及び装置及びコンピュータ制御装置を提供することを目的とする。

【 0 0 0 8 】また、本発明の他の目的は、複数のフラッ

シュROMに対して複数のタスクが効率良く書き込みを与えるフラッシュROM管理方法及び装置を提供することにある。

【 0 0 0 9 】また、本発明の他の目的は、低優先度のタスクで消去処理の処理速度向上のための前処理を行うことを可能とすることにある。

【 0 0 1 0 】

【課題を解決するための手段】上記の目的を達成するための本発明のフラッシュROM管理装置は以下の構成を備える。即ち、複数のタスクが並列的に実行されるシステムにおけるフラッシュROMの管理装置であって、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュROMに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュROMのアクセスを管理する管理手段と、前記複数のタスクの一つによって前記複数の記憶ブロックの一つがアクセスされた場合、そのアクセスされた記憶ブロックについて、他のタスクからのアクセスを禁止する排他手段とを備える。

【 0 0 1 1 】また、好ましくは、前記管理手段は、フラッシュROMへのアクセスがデータの書き込みであった場合、未使用状態の記憶ブロックを検索して、検索された記憶ブロックへデータの書き込みを行い、前記排他手段は、前記未使用状態の記憶ブロックの検索中は、他のタスクから未使用状態の記憶ブロックの検索を行うことを禁止する。複数のタスクによって同じ記憶ブロックが検索されることを防止できるからである。

【 0 0 1 2 】また、好ましくは、前記管理手段は、前記複数のタスクによる複数のフラッシュROMへの書き込みをラウンドロビンスケジューリングによる時分割多重で同時に行う。

【 0 0 1 3 】また、本発明の他の目的を達成するフラッシュROM管理装置は、複数のタスクを優先度を持たせて並列的に実行させることが可能なシステムにおけるフラッシュROMの管理装置であって、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュROMに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュROMのアクセスを管理する管理手段と、状態情報が無効データを示す記憶ブロックに対して消去速度を向上するための前処理を行う前処理手段と、前記前処理手段の実行を前記複数のタスクのうちの低優先度のタスクに割り当てて実行する実行手段とを備える。

【 0 0 1 4 】

【発明の実施の形態】以下に添付の図面を参照して本発明の実施の形態を説明する。

【 0 0 1 5 】【実施形態1】

<カメラシステムの構成>図1は実施形態1におけるカ

メラシステムの構成を表すブロック図である。本カメラシステムは、電子カメラと、これに着脱可能な外部記憶媒体 1 7、PC 通信インターフェース 1 9、及び PC 通信インターフェース 1 9 を介して電子カメラと通信可能に接続されたパーソナルコンピュータ 2 2 から構成される。

【0 0 1 6】1 はレンズであり、2 はレンズ 1 を通った光を電気信号として出力する CCD ユニットである。3 は A/D コンバータであり、CCD ユニット 2 からのアナログ信号をデジタル信号へ変換する。4 は SSG ユニットであり、CCD ユニット 2 と A/D コンバータ 3 に同期信号を供給する。5 は CPU であり、本カメラシステムにおける各種の制御を実現する。

【0 0 1 7】6 は信号処理アクセラレータであり、信号処理を高速に実現する。7 は電池であり、8 は、電池 7 からの電力を電子カメラ全体へ供給するための DC/D コンバータである。9 は電源コントローラユニットであり、DC/D コンバータ 8 をコントロールする。1 0 はパネル操作・表示装置・電源のコントロールを行うマイクロコンピュータである。1 1 はユーザへ各種の情報を表示する表示装置であり、液晶パネル等が用いられる。1 2 はコントロールパネルであり、ユーザが直接操作するリリーススイッチを含む。

【0 0 1 8】1 3 は ROM であり、OS 等のシステムプログラムを格納する。1 4 は DRAM であり、本電子カメラの主記憶である。1 5 はフラッシュ ROM であり、内蔵記憶媒体として使用する。1 6 は PCMCIA カードのインタフェース部、1 7 は ATA ハードディスクなどの外部記憶媒体、1 8 は拡張バスインタフェースである。1 9 は PC 通信インタフェースであり、パーソナルコンピュータ等を接続してデータの授受を行う。2 0 は DMA コントローラ、2 1 はストロブである。また、2 2 はパーソナルコンピュータであり、PC 通信インターフェース 1 9 を介して、電子カメラとの通信を行う。

【0 0 1 9】＜撮影動作＞この電子カメラの撮影時の動作を簡単に説明する。コントロールパネル 1 2 のリリーススイッチをユーザが押すと、CPU 5 がそのことを検出して撮影シーケンスを開始する。以下の動作は全て CPU 5 によるコントロールで行われることを前提とする。

【0 0 2 0】さて、リリーススイッチの押下により、SSG 4 が CCD 2 を駆動する。CCD 2 から出力されるアナログ信号は、A/D コンバータ 3 でデジタル信号へ変換される。A/D コンバータ 3 の出力は、DMA コントローラ 2 0 によって DRAM 1 4 へ DMA 転送される。1 フレーム分の DMA 転送が終了した時点で CPU 5 は、信号処理シーケンスを開始する。

【0 0 2 1】信号処理シーケンスでは、フラッシュ ROM 1 5 から信号処理プログラムを主記憶 (DRAM 1 4) 上に読み出し、主記憶上のデータを信号処理アクセ

ラレータ 6 へ転送し信号処理を行う。但し、信号処理アクセラレータ 6 は信号処理の全てを行うわけではなく、CPU 5 で行う処理の特に時間のかかる処理などを助ける演算回路であり、CPU 5 の処理ソフトウェアと連携して動作する。信号処理の一部または全部が終了すると画像ファイルとしてフラッシュ ROM 1 5 へ記録する。この時記録するファイルフォーマットが圧縮処理を必要とするのであれば圧縮も行う。

【0 0 2 2】信号処理プログラムは、フラッシュ ROM 1 5 の中でファイルシステムが管理するファイルの 1 つである。カメラのプログラムは OS やファイルシステムといっしょに ROM 1 3 に納められている。カメラのプログラムは、特定のファイル名のファイルをプログラムであると認識する。

【0 0 2 3】フラッシュ ROM 1 5 の中でファイルは不連続に配置されている上に、本実施形態のファイルシステムが頻繁に再配置を行うため、フラッシュ ROM 1 5 内の制御プログラムを CPU が直接実行することはできない。従って、主記憶 (DRAM 1 4) に読み出して実行させなければならない。更に、主記憶はメモリマネージャが動的に記憶場所をアロケーションするため、特定のアドレスに格納されることを想定したソフトウェアであってはならない。そのため、本実施形態で信号処理を行うプログラムのファイルは図 4 1 のような形式となっている。

【0 0 2 4】図 4 1 は本実施形態におけるフラッシュ ROM への制御プログラムの格納状態を説明する図である。図 4 1 において、識別コードはファイルがプログラムであることを確認するためのコードである。ファイルは可変長のレコードの集合として表現されている。レコードには、始めに当該レコードに格納されている情報の種類を識別する ID があり、次にそのレコードの大きさを示す値が格納されている。

【0 0 2 5】そして、プログラムのレコードとリロケーション情報のレコードがファイルに格納されている。プログラムコードは例えば図 4 2 のようなデータである。図 4 2 は相対アドレスで表現されたプログラムコードの一例を表す図である。図 4 2 では、0 0 5 0 番地にジャンプ命令があるが、CPU はこの命令を絶対番地へのジャンプ命令と認識する。この命令のオペランドは相対アドレスで表現されている。

【0 0 2 6】図 4 2 のリロケーション情報レコードのデータは図 4 3 のような形式で納められている。即ち、図 4 2 のプログラムの中で、絶対番地へ変換しなければならないデータ (相対アドレス表現になっているデータ) のプログラム番地を示すアドレステーブルがリロケーション情報として格納される。

【0 0 2 7】図 4 1 のファイルを主記憶にロードするための領域を確保すると、ROM 1 3 の OS のメモリマネージャがアドレスを決定する。メモリマネージャのアロ

10

20

30

40

50

ケーションはC言語ではalloc関数に相当する機能である。メモリマネージャがプログラム用に8710番地を割り当てた場合図44のようにプログラムがロードされる。図44は図41のプログラムを主記憶の8710番地へマッピングした場合のプログラムコードを示す図である。ジャンプ命令のオペランドが実際の絶対番地に置き換えられている。この実アドレスへの変換をしながらプログラムを主記憶へ読み出すという作業を行うプログラムはROM13に格納されている。

【0028】以上のように構成することにより、記憶媒体に信号処理ソフトウェアや圧縮ソフトウェアをファイル形式で格納することができる。その結果、カメラが最終ユーザの元へ届いてから、新しい信号処理アルゴリズムや、Windows（商標）のBMP形式やTIFF形式、あるいは将来新たに登場する形式等、多種多様なファイル形式への対応が可能となる。

【0029】以上の様に、本実施形態1における電子カメラは、撮影画像をフラッシュROM15へファイルするものである。

【0030】＜デバイスドライバインタフェース＞図2は、本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。最上位の層がユーザアプリケーション101である。ユーザアプリケーション101は電子カメラの内部で動くソフトウェアであり、ファイルをファイル名でオープンして読み書きした後クローズする。

【0031】ユーザアプリケーション101から直接ファンクションコールによって呼び出されるのがファイルシステムAPI層102である。このファイルシステムAPI層102がドライブ名とファイルシステムを関連付けて管理している。各ドライブ毎にファイルシステムアーキテクチャ層をマウントする用に構成しているため、複数のファイルシステムアーキテクチャを混在させる事が可能となっている。

【0032】ファイルシステムアーキテクチャ層103が実際のファイル管理を行う部分である。最下位の層がブロックデバイス層104である。ファイルシステムアーキテクチャ層103がブロックデバイス層104の提供するサービスを利用してファイル入出力を実現している。このブロックデバイス層104では、データをセクタという単位で管理しており、1セクタは例えば512バイトである。このブロックデバイス層104でデバイスごとの入出力制御の違いと、ヘッドやシリンダなどパラメータの違いを吸収している。このように構成しているため、同時に複数の種類のデバイスを混在させることができる。

【0033】本実施形態の電子カメラでは、特にブロックデバイス層104におけるフラッシュROMの記憶管理方法に特徴を有する。

【0034】図1で示したフラッシュROM11には、

現在いろいろなタイプのものがあるが、大きく分けてフラッシュDISK用に開発されたタイプとパーソナルコンピュータのBIOS用に開発されたタイプがある。前者は消去単位がハードディスクで一般的な512バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュROMは消去単位が例えば64kなどといった大きなブロック単位でしか行えない様になっている。また、PROMの様に書き込み電圧として12V等の電圧が必要なものもある。しかしながら、後者のタイプのフラッシュROMは安価で入手が容易である。本実施形態では、後者の様な特徴を持つフラッシュROMでありながらファイルシステムに対してハードディスク同様のサービスを提供する。

【0035】＜フラッシュROMドライバインタフェース＞一般的にブロックデバイスがファイルシステムへ提供するサービスは以下の2つである。即ち、

(1) ロジカルセクタナンバーで指定したセクタからの読み出し

(2) ロジカルセクタナンバーで指定したセクタからの書き込み

である。そして、これに加えて

(3) ロジカルセクタナンバーで指定したセクタの開放の機能があれば、フラッシュROMのドライバは必要に応じて不要なセクタを消去することが可能となるため、効率良くフラッシュROMを消去することができる。

【0036】(3)に挙げた機能は、通常のDISKでは必要のない機能だが、キャッシュを持ったシステムだと、積極的にキャッシュリストから削除できるので、結果的にキャッシュのヒット率を上げる効果がある。ファイルシステムは、ファイルの消去等で不要となったセクタを(3)の機能を用いてデバイスドライバへ通知する。フラッシュROMの消去は非常に時間がかかる処理だが、CPU時間をほとんど消費しないためバックグラウンド処理で行うのが良い。

【0037】後述の＜FATキャッシュ＞においても説明するが、本実施形態のキャッシュは、新しいデータ（キャッシュ上に無いデータ）をアクセスする場合にキャッシュリストの中で最も古いデータを廃棄する。不要セクタをキャッシュリストの最後へ移動させる（即ち、最も古くアクセスしたデータがキャッシュの後ろへ移動すること）で有効なデータがキャッシュから廃棄される可能性が低くなる。特にコンパイラ等の中間ファイルを多く生成するシステムでは、消去すべき中間ファイルがキャッシュに残っている可能性が高く、上記のキャッシュ管理はヒット率の向上に非常に有効である。

【0038】図3は、デバイスドライバの管理ブロックをC言語で記述した宣言文を示す図である。構造体のNextは、次のデバイスへのリングポインタであり、メモリ中のデバイスを検索する目的で使用される。DevNameは、デバイスの名前として使用される。Ini

tDevは、デバイスの初期化ルーチンへのポインタである。ShutDownは、デバイスのシャットダウンルーチンへのポインタである。ReadSectorは、ロジカルセクタを指定して媒体の内容をバッファへ転送するルーチンへのポインタである。WriteSectorは、ロジカルセクタを指定してバッファの内容を媒体へ転送する（書き込む）プログラムへのポインタである。ReleaseSectorはロジカルセクタを指定して、セクタを解放するルーチンへのポインタである。

【0039】ファイルシステムは、この構造体を仲介してデバイスドライバを利用することになる。固定ディスクやフロッピーディスクの場合、ReleaseSectorには何も仕事をしないプログラムへのポインタが代入されている。または、ディスクキャッシュのキャッシュリストから指定セクタを削除するポインタでもよい。

【0040】＜フラッシュROM管理方法＞フラッシュROMに対するデータ書き込みは、上位層のファイルシステムからセクタ単位で行われる。図4は、フラッシュROM上のセクタ構造の例を示す図である。図4において、151はイレースブロックである。このイレースブロック151は消去の単位であり、フラッシュROMの技術用語ではセクタと呼ばれるものである。しかしながら、ファイルシステムが扱う単位である“論理セクタ”と区別する為に、ここではイレースブロックと呼ぶことにする。

【0041】図4によれば、システム中に複数のフラッシュROM15が搭載されていて、各フラッシュROM15は複数のイレースブロック151によって構成される。更に、各イレースブロック151は消去回数カウンタ152と複数のセクタ153によって構成されている様子をあらわしている。消去回数カウンタ152は、イレースブロック151を消去した回数をカウントする為に用いる部分である。各セクタ153は、管理領域とデータ領域とを有する。管理領域は、論理セクタ番号を表すセクタ番号154と、セクタが有効利用されているかどうかを表わす使用中フラグ155と、セクタとしての利用が終了したことを表わす使用済みフラグ156とで構成される。また、データ領域は、512バイトのデータ部157によって構成されている。

【0042】データ領域と管理領域は、隣り合って配置する必要は無く、図5の様にまとめて管理することも考えられる。図5は、管理領域用データと、データ領域とを分離して格納する構成を表す図である。セクタ番号テーブルには、複数のセクタ153の各セクタ番号154が格納される。また、フラグテーブルには、使用中フラグ155、使用済みフラグ156が格納される。更に、データテーブルには、データ部157の内容が格納される。以上のようなデータ構成をとることも可能である

が、少なくとも管理領域と、これに対応するデータ領域とは同じイレースブロック内に納めるのが好ましい。

【0043】なお、システムは「使用中フラグ155」より「使用済みフラグ156」の方を優先的に評価する。図6は、各フラグの状態に対応した意味を示す図である。図中、FALSEは、消去後の状態と同じ値をとる。使用中フラグ155がTRUEであっても、使用済みフラグ156がFALSEであれば、当該セクタのデータは無効である。

10 【0044】＜フラッシュROMの論理セクタ書き換え＞フラッシュROMはPROM同様、データを書き換える為に、一度消去してから再書き込みをしなければならない。しかも、消去の最少単位が大きく（例えば64kバイト）消去時間が長い（例えば1秒）。そこで上位層のファイルシステムが、特定のセクタを書き換えようとした場合、消去済みの領域へ論理セクタを移動させることで、消去動作をせずに、見かけ上で論理セクタのデータ書き換えを実現する。

20 【0045】図7はセクタの書き換え手順を説明する図である。同図を用いて、8番セクタ（論理セクタ番号が8のセクタ）の書き換えを例にして詳しく説明する。図7中、左側が書き換え前の状態であり（（a）の状態）、右側が書き換え後の状態（（b）の状態）である。また、図7において、管理領域中の数字は論理セクタ番号を表し、（使用中）は使用中フラグ155がTRUEで使用済みフラグ156がFALSEの状態、（使用済）は使用中フラグ155と使用済みフラグ156が共にTRUEの状態を示す。

30 【0046】“セクタ番号8（使用中）”の場所に、8番セクタのデータが格納されている。今、8番セクタがFATやファイルの一部として利用されていて、その内容を変更したい場合に上位層から8番セクタの書き換え要求が発生したとする。書き換え要求が発生すると、フラッシュROMのデバイスドライバは、フラッシュROMの未使用セクタを検索し、その場所を新たな8番セクタの場所としてセクタ番号と更新後のデータを格納し、使用中フラグをTRUEにする。次に、以前8番セクタだったセクタの使用済みフラグをTRUEにする。このような手順で8番セクタのデータの書き換えが実現される。

【0047】＜ガベージコレクション＞以上の様な方法で論理セクタの書き換えを実行していくと、いずれフラッシュROMのほとんどの領域を“使用済セクタ”にしてしまうことになる。そこであるタイミングでフラッシュROMを一旦消去して“使用済セクタ”を“未使用セクタ”へ戻す必要がある。基本的なガベージコレクションの動作を図8を用いて説明する。図8は、本実施形態におけるフラッシュROMのガベージコレクション動作を説明する図である。

50 【0048】図中（A）は、ガベージコレクション前の

状態である。説明を簡単にするために、本例のフラッシュROMはセクタ6個分の大きさのイレースブロックで構成されているものとする。イレースブロック(1)には使用済セクタが3個と使用中セクタが3個あり、消去回数は5回である(消去回数カウンタ152の内容が5である)。イレースブロック(2)には使用中セクタが1個、使用済セクタが1個、未使用セクタが4個あり、消去回数は9回である。この状態からガベージコレクションを開始する。

【0049】まず、調整対象イレースブロックを選定する。ここで調整対象イレースブロックは消去を行う対象としてイレースブロックとなる。調整対象イレースブロックの選定は、使用済セクタをたくさん含むイレースブロックから優先的に選択すると整理効率が良い。しかし使用済セクタを含まない場合を別として消去回数の少ないイレースブロックを優先的に整理対象とする方法を取ればチップ内のイレースブロックを平均的に使用することができ、書き換え耐久を分散させることができる。選択手順の詳細についてはフローチャートを用いて後述する。

【0050】今、イレースブロック(1)が整理対象として選定されたとする。次に、整理対象であるイレースブロック(1)の使用中セクタ(使用中フラグがTRUEで、使用済みフラグがFALSEのセクタ)を他のイレースブロックに移動させる。使用中セクタの移動手順は、セクタの書き換え時と同様に、他のイレースブロック中の未使用セクタを検索して、使用中セクタ内のデータ領域と管理領域の内容をコピーし、移動元の使用中セクタの使用済みフラグをTRUEにする。なお、未使用セクタが無い場合の処理は、後で述べる。

【0051】図8の(B)は、イレースブロック(1)の使用中セクタをすべてイレースブロック(2)へ移動させた状態である。この結果、イレースブロック(1)には、使用済セクタしか存在しないことになる。

【0052】次に、使用済セクタだけで構成されているイレースブロックを検索する。ここで、検索を行うのは、通常の書き換え動作の際に偶然イレースブロック内のセクタが全て使用済セクタとなっている場合があるからである。続いて検索されたイレースブロックに対して消去を行う。消去には時間がかかるが、複数のイレースブロックを同時に消去できるため、できるだけ一度に複数のイレースブロックを消去するのが良い。消去が終了すると消去回数カウンタへ消去前の値+1したものを書く。これでガベージコレクション完了である。

【0053】図8の(C)がガベージコレクション終了時の状態である。イレースブロックをできるだけ同時に消去した方が効率が良いため、使用済セクタと未使用セクタがある限りたくさんのイレースブロックを同時に調整すると良い。極端に消去回数カウンタの値が他のイレースブロックより少ないものがあれば、使用済セクタを

含んでいなくても整理さえすれば、書き換え耐久の分散を図れる。また、一度整理するとデータの配列が変わる為、書き換え耐久分散のきっかけとなる。

【0054】<未使用セクタがない場合>次にシステム中に使用済セクタが有るにもかかわらず未使用セクタが全く無くなってしまった場合のガベージコレクション手順を図9を使って説明する。図9は、未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【0055】まず、上述した基本的なガベージコレクション手順に従い、イレースブロック(1)を整理対象として選択する。次にイレースブロック(1)の使用中セクタを移動する為の未使用セクタを検索する。未使用セクタがある場合は、上述の基本的ガベージコレクションと同様にセクタの移動を行う。

【0056】一方、検索の結果、未使用セクタが無ければ、DRAM14のヒープエリアからデータの退避に必要な大きさのメモリブロックをアロケーションする。そして調整対象イレースブロック内の使用中セクタをDRAM14へコピーする。この場合は、フラッシュROMの別の領域へセクタを移動する場合と違い、元のセクタの使用済フラグをTRUEにしない。なぜならこの時点で電子カメラの電池7が外れるなどの事故が起こった場合に、DRAM内のデータが消滅してしまい、データの修復ができなくなるからである。図9の(B)は、DRAM14の領域へコピーされたセクタを表現している。調整対象のイレースブロックの使用中セクタをすべて退避出来たら、その調整対象のイレースブロックを選択して消去する(図9の(C)参照)。消去が終わった後は未使用セクタがたくさん出来ているはずである。よって、次に未使用領域を検索してDRAM14へ待避してあったデータを復元する。図9の(D)は、ガベージコレクションが完了した状態を示している。

【0057】上記の手順でガベージコレクションをした場合でも、イレースブロックを消去してからデータを復元するまでの間に電子カメラの電池7が外れるなどの事故が起こったらデータの修復をすることはできない。つまり、セクタのデータをDRAM14に退避する方法はできる限り取らない方が、よりシステムの安全性を保つことができる。一回、DRAMを使ってガベージコレクションを行えば未使用セクタができる。したがって、1度DRAM14を使ったガベージコレクションを行い、その後に通常のガベージコレクションを行えば、消去時間は余分にかかるが安全性を高めることができる。逆にDRAM14への退避を積極的に行う(例えばヒープ領域がある限り退避する)と、同時に整理できるイレースブロックが増える為効率を上げることができる。従って、安全性と効率のどちらを優先するかを指定できるように構成してもよい。またシステムの電源が電池7より供給されている場合は安全性優先、ACアダプタから供

給されている場合は効率優先に自動的に切り替わるように構成してもよい。本処理については図 3 6 を参照して後述する。

【 0 0 5 8 】 < イレースブロックを余分に 1 つ用意 > 残り容量が極端に少なくなるとガベージコレクションが多発してシステムのパフォーマンスが極端に落ちる。総論理セクタ分を格納できるイレースブロック数よりも 1 つだけ余分にイレースブロックを使用すれば、そのような事態を避けることが可能である。仮に、1 イレースブロックあたり 1 2 7 セクタ格納できるとして、全てのセクタが使用中となった場合、同じ 1 セクタを 1 0 回書換える場合を例にすると、余分イレースブロックがなければ、1 0 回の消去と 1 2 7 0 セクタの書き込みが発生する。しかし、イレースブロックを余分に 1 つ用意しておけば、1 0 セクタの書き込みのみが発生しない。

【 0 0 5 9 】 よって、本実施形態では、イレースブロックの数はチップの構成で決まるので、最低 1 つのイレースブロックが余る様な総論理セクタ数を設計する。

【 0 0 6 0 】 < ガベージコレクションのタイミング > ガベージコレクションは消去動作を伴うために非常に時間がかかる。そのためガベージコレクションをいつ行うかによってカメラの使い勝手を左右することとなる。例えば、セルフタイマなどの数秒間撮影しなくても良い時にガベージコレクションを行えばユーザがストレスを感じることがない。

【 0 0 6 1 】 < RAM 上の記憶場所管理 > フラッシュ ROM 1 5 上ではセクタ番号と実際の記憶場所が関連していないために特定のセクタを読み書きする為にフラッシュ ROM 1 5 を検索しなければならない。そこでシステムがリブートする際に、フラッシュ ROM 1 5 における各セクタの格納アドレスを示す記憶場所管理テーブルを DRAM 1 1 4 上に作成しておくこと、フラッシュ ROM 1 5 に対して高速なデータの読み書きを実現できる。一度、記憶場所管理テーブルを作成すれば、フラッシュ ROM 1 5 に対するセクタの書き込みやガベージコレクションによって記憶場所に変更が生じた場合に限って記憶場所管理テーブルの記憶位置を更新するだけで常に正しい記憶場所管理テーブルを維持することが可能である。

【 0 0 6 2 】 図 1 0 は、DRAM 上に作成された記憶場所管理テーブルを説明する図である。図中、右側に RAM 上の作成した記憶場所管理テーブル 1 4 0 を示した。0 セクタと 4 セクタは記憶場所不在を意味する値 (NULL) が入っている。これらは、フォーマット後そのセクタに対する書き込みが全く無かったか、もしくは、ファイルシステムが開放したセクタである。

【 0 0 6 3 】 ファイルシステムがファイルの消去などで不要となったセクタを解放する命令をドライバに出した場合のデバイスドライバの動作は次のようになる。まず、DRAM 1 4 上の記憶場所管理テーブル 1 4 0 の指定されたセクタのポインタを参照してフラッシュ ROM

1 5 上の現在使用中の該当するセクタを探し出す。そして、当該セクタの使用済フラグを TRUE にし、DRAM 1 4 上の記憶場所管理テーブル 1 4 0 の指定セクタのポインタへ不在値 (NULL) を代入する。

【 0 0 6 4 】 なお、ガベージコレクションの為にセクタの内容を DRAM 1 4 へ待避している場合は、記憶場所として DRAM へのポインタが代入されている。また、同一論理セクタに対する同時操作を禁止する為のロック変数もテーブルに納めることが望ましい。

10 【 0 0 6 5 】 < MS-DOS のファイル復元 > 本電子カメラとパーソナルコンピュータ 2 2 で、記憶媒体上のデータ交換が出来ると都合が良い。本実施形態で説明したフラッシュ ROM 管理方式を使用して、現在パーソナルコンピュータで普及している MS-DOS (商標) と互換性があるファイルシステムを実装することができる。MS-DOS には一度消去したファイルを復元するユーティリティが付属している。ところが、本実施形態ではフラッシュ ROM の消去効率を向上させる為に、消去したセクタのデータ部を失ってしまう様な構成となっている。カメラで消去した媒体をパーソナルコンピュータで復元する事が原理的にできない構成になっているのである。

【 0 0 6 6 】 パーソナルコンピュータでのファイル復元機能を禁止できれば、このような事故を防ぐことができる。本実施形態では、MS-DOS がファイル復元の時に使用するデータを破壊することでファイル復元機能を禁止する。これをいかに説明する。

30 【 0 0 6 7 】 MS-DOS (商標) で、ファイルを消去するとディレクトリに空きスロットができる。ディレクトリにはファイル名 / タイムスタンプ / 最初のクラスタなどの情報が格納されている。図 4 5 はディレクトリスロットの特徴を表す図である。ディレクトリスロットの最後には、リストの最後であることを示す EndOfDir が格納されている。

【 0 0 6 8 】 今、File B を削除すると、ファイル名の先頭が削除を表すシンボルに置き換えられ、FAT のクラスタチェーンが消去される。この様子を図 4 6 に示す。

【 0 0 6 9 】 アンデリートプログラムは、2 番目のスロットに残った情報を元に、ファイルの復元を試みる。逆にこの情報がなければ、ファイルの復元を防止できる。

【 0 0 7 0 】 図 4 7 は本実施形態の DOS 互換ファイルシステムでファイルを消去した後の状態を表している。本実施形態では、ディレクトリエントリテーブルの最後に格納されているファイルを消去したいファイルのエントリに上書きし、ディレクトリエントリテーブルの最後のファイルだった部分に EndOfDir を上書きするように構成する。こうすることにより、ファイル復元機能によるファイルの復元を防止できる。

50 【 0 0 7 1 】 なお、ファイルの消去時には MS-DOS と同様にセクタのデータをそのまま残しておき (セクタ

の開放を行わない)、ガベージコレクション時にまとめてFATとデータの関係を参照してしながら不要部分を消去する方法もある。

【0072】<バックグラウンドで前処理>あるフラッシュROMでは、消去前のデータが“0”になっている方が高速に消去処理できる。フラッシュROMの消去完了の確認は、データ書き込み時と同様にデータポーリングによって行われる。従って、このようなフラッシュROMを使う場合は、バックグラウンド処理で“使用済”となったセクタのデータを0に書き換える「前処理」を行うことで性能を向上させることができる。最も低いプライオリティのタスクとして実行するようにしておけば、スループットの低下にはつながらない。

【0073】この前処理バックグラウンドでの“前処理済セクタ”管理の為にフラグを用意しておけば前処理の効率を上げることができる。

【0074】そのために、セクタのとりうる状態として、「未使用」「使用中」「使用済」に加えて「前処理済」の4つの状態を表示できる管理フラグをフラッシュROMのセクタ内部へ用意すると効率が良い。

【0075】図38は、本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。同図において、ステップS2501にて、使用済でかつ前処理の済んでいないセクタを抽出する。これは、セクタ内の管理フラグが「使用済」となっていて、かつ「前処理済」となっていないセクタを抽出することで実現できる。ステップS2502において、抽出されたセクタに対してデータ「0」を上書きを開始する。ステップS2503では当該セクタについて前処理を終了したか否かを判断する。フラッシュROMへの書き込みは1バイト単位であるので、1セクタ分のバイト数の書き込みが必要となる。当該セクタに対する前処理が終了していなければステップS2504へ進み、他のタスクへ制御を移す。

【0076】上述したように本処理は最もプライオリティの低いタスクで行われるので、CPU5がアイドル状態となったときに再び本処理が実行される。この場合処理はステップS2503へ戻る。この時点で、前回の書き込みが終了していなければそのまま他のタスクへ処理を移行する。

【0077】以上のようにして当該セクタの全バイトに対して「0」の書き込みを終えると、ステップS2503からステップS2505へ進み、当該セクタの管理フラグを、「前処理済」を示す状態にセットする。そして、引き続き、他のセクタについて前処理を行うために、ステップS2501へ戻る。

【0078】<FATキャッシュ>本システムでは、書き込み発生の度に記憶場所を変更し、その度に「未使用セクタ」が発生する。そこで、使用頻度の多い部分を特に優先的にバッファリングするキャッシュがあればト

タルの書き込み頻度が激減する事が予想される。キャッシュとして用意するメモリは多ければ多いほど良いが、システムのメモリには限界がある。

【0079】本来使用頻度の高いセクタのデータは、キャッシュ中に存在する確率も高いが、使用頻度の低いセクタを大量に読み書きした場合、当然キャッシュから吐き出されることになる。

【0080】そこで、ファイルシステムが管理する管理領域を優先的にキャッシングする様に構成すれば、スループットの向上を期待できる。なぜならファイルシステムの管理領域は頻繁に更新されているからである。

【0081】パーソナルコンピュータで普及しているMS-DOSのFATシステムの場合、720kや1.4Mといったフォーマット形式では1クラスタが1セクタで構成されている為、シーケンシャルにファイルを読む場合でも2回に1回はFATを読まなければならない。ファイルを書く場合は、さらにたくさんのFATアクセスが発生する。このため、システム中にたくさんのファイルがオープンされるとキャッシュのヒット率が落ちてしまう。

【0082】アプリケーションソフトウェアにもよるが、FATシステムにおいてFATのみを対処にしたキャッシュは、DISK全体を対象にしたキャッシュに対して1/2のメモリで同等のヒット率を確保できる。図11はキャッシュソフトウェアの階層的な位置付けを表す図である。キャッシュのソフトウェアは図11の様にファイルシステムとフラッシュROMの中間的な場所となる。

【0083】図12はキャッシュの主記憶上のデータ構造を表わす図である。片方向線形リスト構造でバッファ全体を管理している。検索方向順にデータが古くなっている。論理セクタ番号が12, 11, 6, 5の順番でアクセスすれば、図12に示されるような順番となる。また、各セクタには、変更フラグが設けられており、キャッシュ上でデータの更新があった場合、変更フラグがFALSEからTRUEに変化する。このようなFATキャッシュの読み出し手順、及び書き込み手順を図13、14を参照して説明する。図13はFATキャッシュの読み出し手順を表すフローチャートである。図14はFATキャッシュの書き込み手順を表すフローチャートである。

【0084】図13において、ステップS1501でNセクタの読み出しを開始する。ステップS1502でNセクタがFATかどうかを判断する。FATでなければ、ステップS1509でフラッシュROM15からデータを読み出す。

【0085】一方、ステップS1502でNセクタがFATならステップS1503へ進み、キャッシュリストを検索する。ここでは、図12で説明した片方向線形リストを検索することになる。キャッシュ中にNセクタが

10

20

30

40

50

存在すればステップ S 1 5 0 7 へ進み、Nセクタのバッファからデータを読み出す。

【0086】また、ステップ S 1 5 0 3 で Nセクタがキャッシュリスト中に存在しなかった場合は、ステップ S 1 5 0 4 へ分岐し、最も長くアクセスされていないセクタのデータ（図 1 2 ではセクタ番号 1 2 のデータ）の吐き出しを行う。まず、ステップ S 1 5 0 4 では、キャッシュリストの最後の項の変更フラグを判断する。もし変更フラグが TRUE なら、ステップ S 1 5 0 5 へ進み、変更内容をフラッシュ ROM 1 5 へ書き込む。変更が無い（変更フラグが FALSE の場合）なら、そのままステップ S 1 5 0 6 へ制御を移す。読み出し手順の中で書き込みを行うのは奇妙に思うかもしれないが、バッファがキャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【0087】ステップ S 1 5 0 6 でフラッシュ ROM 1 5 からリスト最後のバッファへ Nセクタの内容を読み出す。ステップ S 1 5 0 7 で Nセクタのバッファからデータを読み出す。ステップ S 1 5 0 8 で Nセクタのバッファをキャッシュリストの先頭へ移動させる。これは、図 1 2 において、各セクタが有する「次のバッファ」（次のバッファを示すアドレス）の値を変更することで達成される。FAT キャッシュへのアクセスが行われる度にステップ S 1 5 0 8 の動作が繰り返されることで、自然にアクセスされないバッファがリストの先頭から最後に向かってシフトしていく。よって、ステップ S 1 5 0 4 でリスト最後のバッファを選ぶのは、最も古いバッファを吐き出す為である。

【0088】次に図 1 4 を参照して書き込み手順を説明する。

【0089】ステップ S 1 6 0 0 で Nセクタの書き込みを開始する。ステップ S 1 6 0 1 では、Nセクタが FAT かどうかを判断する。FAT でなければ、ステップ S 1 6 0 8 へ進み、フラッシュ ROM 1 5 へのデータの書き込みを実行する。

【0090】一方、ステップ S 1 6 0 1 で Nセクタが FAT ならば、ステップ S 1 6 0 2 へ進み、キャッシュリストを検索する。キャッシュ中に Nセクタが存在すればステップ S 1 6 0 6 へ進み、Nセクタのバッファヘデータの書き込みを実行する。

【0091】また、ステップ S 1 6 0 2 で Nセクタがキャッシュリスト中に存在しなかった場合は、ステップ S 1 6 0 3 へ分岐し、バッファから最も長くアクセスされていないセクタの吐き出しを行うとともに、Nセクタをキャッシュに登録する。まず、ステップ S 1 6 0 3 でキャッシュリストの最後の項の変更フラグを判断する。もし変更フラグが TRUE なら、ステップ S 1 6 0 4 で変更内容をフラッシュ ROM へ書き込み、ステップ S 1 6 0 5 へ進む。また、変更が無いなら（変更フラグが FALSE なら）そのままステップ S 1 6 0 5 へ制御を移

す。ステップ S 1 6 0 5 では、キャッシュリストの最後の項を Nセクタとする。その後、ステップ S 1 6 0 6 で、Nセクタのバッファヘデータの書き込みを実行する。

【0092】その後、ステップ S 1 6 0 7 で N、セクタのバッファをキャッシュリストの先頭へ移動させる。書き込み手順の中でフラッシュ ROM への書き込みを行わないのは奇妙に思うかもしれないが、キャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【0093】また、ステップ S 1 5 0 1 及びステップ S 1 6 0 1 における、FAT の判断であるが、IC カード等の完全に上位層（ファイルシステム）の情報を共有できないシステムでも、書き込みデータの内容を解析することで FAT 領域の場所を特定できる。なぜならば、論理セクタ O に相当する部分に FAT の位置等の情報が格納されていることが決まっているからである。

【0094】＜フラッシュ ROM への 1 バイトの書き込み＞フラッシュ ROM 1 5 に対する全ての（管理領域を含む）読み書きは、最終的に 1 バイトの読み書き命令によって実行される。フラッシュ ROM 1 5 の書き込みには、通常の PROM 同様の時間がかかる。1 バイトの書き込みが終了するまでは、同じチップへの書き込みはできない。書き込み終了信号として信号線が用意されているチップと特別な信号が用意されていないチップがある。後者の場合は、データポーリングと言う手法で書き込み終了を確認しなければならない。データポーリングとは、ペリファイに非常によく似た方法で、書き込みデータと読み出しデータが一致するまで待つビジー制御方法である。

【0095】信号線によって書き込み終了を知ることが出来る場合は、CPU 5 への割り込みと併用して書き込み待ち中の CPU タイムを別のタスクへ割り当てることができる。

【0096】上述のように、信号線が無いチップの場合は、データポーリングを行わなければならない。データ書き込みの効率をあげる為にはいくつものチップに対してパイプライン的に書き込みを行い、データポーリング時間のロスを押さえなければならない。そのため、1 バイトの書き込みが完了する前に次の動作へ制御を移す必要がある。新たな読み書きを行う前に以前の書き込みが完了しているかどうかを確認するのが良い。図 1 5 は、その様子を C 言語で表現したものである。

【0097】図 1 5 の 1 行目は、データ書き込みを行う関数の入り口である。最初の引数は最後に書き込んだアドレスとデータを保存するための構造体へのポインタ、第 2 の引数は書き込むアドレス、第 2 の引数は書き込むデータである。

【0098】3 行目では、最後に書き込んだアドレスを参照してチップに書かれたデータと最後に書いたデータを比較して、両者が一致するまでループを実行する。こ

れがデータポーリングである。前回の書き込みが完了するとこのループから抜け出す。

【0099】4行目で新しいアドレスへDataを書き込む。5行目と6行目で、今回書いたアドレスとデータを保存する。この情報は、次のデータポーリングで利用される。

【0100】リスト7行目のRotateRdyQueueは、自タスクの次に実行されるべき同一プライオリティの実行可能状態のタスクへCPUを譲るオペレーティングシステムのシステムコールである。

【0101】9行目は読み出し関数の入り口である。第1の引数はアドレスとデータを保存するための構造体へのポインタ、第2の引数は読み出すアドレスである。この関数は上位のプログラムに対して第2の引数で指定されたアドレスに格納されたデータを返す。

【0102】11行目では、もし読み出そうとしたアドレスが最後に書き込んだアドレスなら戻す値は最後に書き込んだデータなので構造体の中に保存された情報を返す。12行目は3行目と同じようなデータポーリングである。データポーリングに成功しないと同一チップの別のアドレスを読むことができない。データポーリングが

終わって13行目で指定したアドレスの内容を戻している。

【0103】1チップへの書き込みを以上の様な構成にしておけば、チップ数と書き込みタスクを増やすだけで確実に見かけ上の書き込み速度を向上させることができる。また、全体のスループットを上げる為にわざとチップ数分のセクタバッファを用意（2チップなら2セクタ）して書き込む内容がバッファに溜まるまで処理しないようにすると効果がある。

【0104】図15のプログラムの特徴的なところは、データポーリングをデータ書き込み直後に行うのではなく、次の書き込みの前に行うことである。そのために前回書いたアドレスとデータを保存しておくRAM領域をチップごとに確保し構造体「struct DEV」として格納しているのである。

【0105】図39は、本実施形態におけるフラッシュROMへの1バイトデータの書き込み手順を表すフローチャートである。本フローチャートは、1つのフラッシュROMチップへの書き込みの制御手順を示している。ステップS2601では、前回の書き込み処理が完了したか否かを判断する。前回の書き込み処理が終了していなければステップS2604へ進み、そのまま他のタスクへ制御を移す。

【0106】一方、前回の書き込み処理が終了していれば、次の書き込みデータを準備し、これをDRAM14へ保存する。上述のステップS2601における書き込み終了の判断は、フラッシュROMに書き込まれたデータと、このステップS2602で保持されたデータとの比較によって行われる。

【0107】続いて、ステップS2603において、データの書き込みを開始する。以上のような処理によれば、複数のフラッシュROMチップに対して、複数のタスクで書き込みを行うような場合に、いわゆるラウンドロビン方式を適用した書き込み処理が可能となり、複数のROMチップに対して効率良くデータの書き込みが行える。なお、マルチタスクの管理プログラムは、上述のROM13に格納されている。組み込みタイプのリアルタイムOSとしては、VxWorks（商標）やpSOS（商標）等が市販されており、ROM13にこれらのようなリアルタイムOSが格納されている。

【0108】＜フラッシュROM書き込み電源の共有化＞データの書き込みや消去の際にPROM同様に12V等の特別な書き込み電圧を必要とするチップや、書き込み電圧を与えることで書き込みが高速になるチップがある。この様なチップを使用する場合に専用のDC/DCコンバータ等の電圧発生部を設けると電子カメラのコストアップにつながる。ところが、従来よりカメラにはストロボの充電や、機構部分やCCDの駆動等、特別な電圧が必要な部分がありDC/DCコンバータ等を搭載している。そこで、フラッシュROMの書き込み電圧とストロボ充電やメカ駆動を時分割多重で行うことで、少容量のDC/DCコンバータでシステムを構築でき、システムのコストを押さえることができる。

【0109】図16は、DC/DCコンバータの出力容量を越えない様に電源を管理するプログラムをC言語で表現したものである。Line1～6が1ステップのズームアップ関数で、Line7～13がフラッシュROMへ1セクタ書き込む書き込み関数である。ズームアップ関数はLine3でDC/DCコンバータの資源管理用のセマフォ“SemDCDC”を獲得して、モータを1ステップ動かす関数をLine4で呼び出す。モータ駆動が終わるとDC/DCコンバータの資源管理用のセマフォ“SemDCDC”を開放する。セマフォはマルチタスクのオペレーティングシステムで資源を管理する為の一般的な方法であり、多くのオペレーティングシステムがシステムコールとして用意している。

【0110】即ち、Line3で既に“SemDCDC”が他のタスクによって使用されていたとすると、他のタスクがセマフォ“SemDCDC”を開放するまでズームアップをしようとしたタスクの実行が保留される。

【0111】書き込み関数はLine9でセマフォ“SemDCDC”を獲得し、フラッシュROMへ1セクタのデータを書き込む。Line11でデータポーリングを行い最後の書き込みが終了したことを確認したら、Line12でセマフォ“SemDCDC”を開放する。このようにプログラムを構成すれば、ズームアップとフラッシュROMの書き込みを同時に行うことは無くなる。ズームは1ステップ単位であり、書き込みはセクタ

単位なので非常に短い保留時間の後に必ず電源を獲得できる。

【0112】図16について更に説明すると、図16のLine1はズームアップする関数の入り口である。本関数には引数はない。Line3で電源の使用権利として宣言したSemDCDCの権利の一つを獲得する。この時、使用権利が1つもなければこの関数を呼び出したタスクの実行は保留される。電源の使用権利を別のタスクが解放すればZoomUpを呼び出したタスクが再び実行可能状態に戻る。そして、Line4のモーターを動かす関数を呼び出すことができる。そして、Line5で、電源使用権利を返却してこの関数の仕事は終了する。Line7は1セクタのデータをEEPROMに書き込む関数の入り口であり、Line9で電源利用権利を獲得してLine12で返却している。

【0113】図40は、上述した電源の共有手順を説明するためのフローチャートである。同図において、ステップS1701で、電源コントローラ9によるDC/D
Cコンバータ8の出力電力の供給が解放されたか否かを判断する。ステップS1702では、電源確保のための指示の内容を解析し、この指示結果に従って、ステップS1703、1705、1707、1709のいずれかに分岐する。

【0114】指示の内容が、CCD駆動電力の供給であれば、ステップS1703へ進み、CCD2に対してCCD駆動のための電力を供給する。そして、ステップS1704にて、CCD駆動の終了（即ち撮影動作の終了）を検出すると、ステップS1711へ進み、電源の解放を行う。また、ストロボの充電要求であれば、ステップS1705へ進み、電源コントローラ9に対してストロボ21に対する充電電力を提供させる。そして、ステップS1706でストロボの充電を完了したら、ステップS1711へ進み、電源の解放を行う。なお、充電の電力供給は、所定時間の充電を行う毎に他の電源供給のために電源を解放する。即ち、ストロボ21への充電を管理するプログラムは別個に所定のタスクに存在し、充電の完了はそのタスクによって管理される。

【0115】指示の内容が、ズーム機構の駆動であれば、ステップS1707へ進み、ズーム機構の駆動系（不図示）へ電力供給を行う。そしてステップS1708で、1ステップのズーム動作を終えたらステップS1711へ進み、電源を解放する。更に、指示の内容がフラッシュROMへの書き込みであれば、ステップS1709へ進み、フラッシュROM15への書き込み電力を供給する。1セクタ分の書き込みが終えたら、ステップS1710からステップS1711へ進み、電源を解放する。

【0116】なお、ステップS1704、1706、1708、1710において、各動作の終了を待つが、この待ちループにおいて、他のタスクへの制御が移り、マルチタスク処理が遂行される。この管理処理は、各タ

クから随時起動が可能であり、複数のタスクで同時に起動される可能性もある為、ステップS1701で電源解放のチェックを行っている。

【0117】以上の図40のフローチャートによれば時分割で電源を利用することが可能となる。しかしながら、すべてのシステム（CCD/ストロボ/ズーム/フラッシュROM）が依存しあった1つのプログラムである。このようなソフトウェアを開発すると、開発/デバッグ/メンテナンスのコストが大きくなり、拡張性や柔軟性を保つのが難しくなる。

【0118】そこで、電源を1つの資源に見立ててOSの提供する資源管理機能を用いることで開発効率を向上させることができる。そこで上述のセマフォによる資源管理を行う。即ち、CCDの駆動部、ストロボの駆動部、ズームの駆動部、フラッシュROMの駆動部のそれぞれの制御プログラムが、電源という資源（セマフォ）を獲得、解放することで、時分割された電源の割当てが行える。

【0119】図48は本実施形態による電源の時分割利用を説明する図である。同図に示されるように、あるタスクA（例えばCCD）によって電源要求が発生したとき、電源セマフォが解放された状態にあれば、そのセマフォを獲得して、電源を占有する（ステップS2001～S2003）。続いてステップS2004において、当該電源よりの電力供給を得て所定の処理を行うと、ステップS2005へ進んでセマフォを解放する。

【0120】一方タスクAより遅れて電源獲得を要求したタスクBでは、ステップS2011における電源要求ではセマフォを獲得できず、ステップS2012により、セマフォの解放待ちとなる。そして、タスクAよりセマフォが解放されると、このセマフォをタスクBが獲得して、電源を占有する（ステップS2013）。その後タスクBで所定の処理を実行し（ステップS2014）、電源を解放する（ステップS2015）。

【0121】以上のようなセマフォによる電源資源の管理により、電源の時分割利用が可能となる。

【0122】なお、図48によれば、電源資源の利用権利を示すセマフォが一つしかないが、複数個のセマフォが存在するようにしても良いことは言うまでもない。

【0123】＜実施形態の電子カメラの動作説明＞図17は、本実施形態のリポートからサービスの開始までの動作手順を表わすフローチャートである。ステップS101でシステムがリポートすると、ステップS102でフラッシュROM15の管理領域をスキャンし、DRAM14上に記憶場所管理テーブル140を作成する。また、この処理と並行して、DRAM14上の未使用セクタカウンタ、使用済セクタカウンタ、使用中セクタカウンタへ、それぞれの状態に対応するセクタがいくつ有るかを数え、セットする。このカウンタは、後にフラッシュROM15に対して操作を行ったときに更新され、記

憶効率を判断するのに用いられる。その後、ステップ S 1 0 3 へ進み、各種のサービスを開始する。

【0124】図 1 8 は、指定セクタの読み出しサービスの手順を表わすフローチャートである。まず、ステップ S 2 0 1 で N セクタの読み出しを開始する。ステップ S 2 0 2 では、N セクタをロックする。セクタのロックはロック変数を使って行う。このロック変数は、記憶場所管理テーブル 1 4 0 で各セクタの記憶場所とともに管理される。ステップ S 2 0 2 では、セクタが既に他のタスクによってロックされている場合、他のタスクによってアンロックされるのを待ち、他のタスクによってアンロックされた後で当該セクタのロックを行う。ロックしたセクタはステップ S 2 0 6 でアンロックするまでの間、自タスクによって占有することが出来る。

【0125】ステップ S 2 0 2 で論理セクタをロックすると、ステップ S 2 0 3 で記憶場所管理テーブルを参照して、当該セクタに有効なデータ記憶されているかどうかを確認する。有効なデータが記録されて無ければ、ステップ S 2 0 4 へ分岐する。ステップ S 2 0 4 では、ダミーのデータ（例えば全部 0 など）をセクタの内容として読み出す。ステップ S 2 0 3 で有効なデータが格納されていると判断された場合は、ステップ S 2 0 5 へ分岐する。ステップ S 2 0 5 では記録場所管理テーブルの値を元にフラッシュ ROM（または主記憶）からデータを読み出す。

【0126】ここで、ガベージコレクションを実行中で N セクタが主記憶（DRAM 1 4）へ退避されていた場合は、記憶場所管理テーブルのポインタは主記憶をポインティングとしている。また、図中点線で囲んだ部分は N セクタを占有している期間である。この様なロック機構によって 1 つのセクタ操作の安全性を保証している為、ガベージコレクションの途中でも操作中でないセクタを自由に読み出すことが可能となっている。

【0127】図 1 9 は論理セクタの書き込みサービスの手順を表わすフローチャートである。ステップ S 3 0 1 で N セクタの書き込みを開始する。ステップ S 3 0 2 でステップ S 2 0 2 と同様に、論理セクタのロックを行う。

【0128】次に、ステップ S 3 0 3 で、記憶場所管理テーブルを検索して、N セクタに有効なデータが記録されているかどうかを判断する。有効なデータが記録されていればステップ S 3 0 4 へ、記録されていなければステップ S 3 0 5 へそれぞれ分岐する。ステップ S 3 0 4 では、それまで有効なデータとして記録されていたフラッシュ ROM（または主記憶）のデータを破棄する。ステップ S 3 0 4 におけるデータ破棄の処理は、図 2 1 のフローチャートを用いて詳しく説明を加える。ステップ S 3 0 4 の後ステップ S 3 0 5 へ制御が移る。

【0129】ステップ S 3 0 5 では、フラッシュ ROM 1 5 において N セクタを書き込むための記憶領域を獲得

する。ステップ S 3 0 5 における記憶領域の獲得手順は図 2 3 を用いて詳しく説明を加える。ステップ S 3 0 5 で正常に記憶領域の獲得に成功すれば、ステップ S 3 0 8 へ制御を移す。ステップ S 3 0 8 では獲得したフラッシュ ROM 1 5 の領域へ N セクタのデータを書き込む。

【0130】一方、ステップ S 3 0 5 でフラッシュ ROM に記憶場所が無い場合、即ち記憶領域の獲得に失敗した場合はステップ S 3 0 6 へ分岐する。ステップ S 3 0 6 はデータの退避用に主記憶を獲得する。主記憶の領域確保はオペレーティングシステムが提供するメモリ管理機能によって行う。これは C 言語で `alloc` 関数に相当する機能である。そして確保した領域を片方向線形リスト構造によって管理する。

【0131】図 2 0 は主記憶上に獲得した退避データリストの様子である。（a）は退避データリストにデータが無い状態であり、リストには、`END_OF_LIST` が代入されている。（b）は退避データリストに、セクタ番号 3, 2 0, 2 2 1 の各セクタの内容が退避されている状態である。

【0132】ステップ S 3 0 9 で、記録場所管理テーブルを更新する。ここで、記録したフラッシュ ROM（または主記憶）へのポインタが代入される。ステップ S 3 1 0 で論理セクタのアンロックを行う。図中点線で囲まれた期間その論理セクタを占有できる。ステップ S 3 1 1 で記憶効率の評価を行う。記憶効率の評価手順については、図 2 1 のフローチャートを用いて詳しく説明を加える。記憶効率の評価の結果、記憶効率が悪化した場合は、ステップ S 3 1 2 へ制御を移す。ステップ S 3 1 2 では上述したガベージコレクションを行う。ガベージコレクションについては、図 2 4 のフローチャートを参照して詳しく説明を加える。ステップ S 3 1 3 で N セクタの書き込みが終了してメインのルーチンへ復帰する。

【0133】なお、記憶場所管理テーブルに納められるのは、記憶場所のポインタ（バス空間上のアドレス）である。図 2 0 の（b）の主記憶に待避されたデータの「次のデータへのポインタ」の次のフィールド（図中ではすぐ下に示されている）からは、図 1 0 の左側にあるフラッシュ ROM 上のデータ構造と互換性がある。記憶場所管理テーブルに納められるのはこの互換部分へのポインタである。このように構成することにより、データの読み出しプログラム側でフラッシュ ROM と主記憶を単一のアルゴリズムで扱うことが可能となる。

【0134】次に、指定されたセクタの記憶を破棄する手順（上述のステップ S 3 0 4）を説明する。図 2 1 は、記憶を破棄する手順を表わすフローチャートである。

【0135】ステップ S 4 0 1 で指定領域の記憶破棄を開始する。ステップ S 4 0 2 では、指定されたセクタを記憶する領域が主記憶上にあるかどうかを判断する。主記憶上にあるならステップ S 4 0 5 へ分岐する。ステッ

プS405で待避セクタリスト（本例では、図20で示した片方向線形リスト）から指定領域を削除する。

【0136】片方向線形リストからの指定領域の削除手順は、まずリストの先頭から検索方向順にリストをたどり、ポインタが自分をポインティングしている項を検出する。そして、この検出された項のポインタに現在自分がポインティングとしている値を代入することで実現する。そして、ステップS406で、リストから削除した主記憶領域をオペレーティングシステムへ返却する。オペレーティングシステムへの記憶領域の返却はC言語のfree関数に相当する機能である。

【0137】一方、ステップS402で指定された領域が主記憶上でない（すなわちフラッシュROM上）ならステップS403へ分岐する。ステップS403では、指定されたフラッシュROM上のセクタの管理フラグを“使用済”へ変更する。これは、使用済みフラグをTRUEにセットすることで達成される。ステップS404では主記憶上の未使用セクタカウンタの値を1つ減少させる。ステップS407で復帰する。

【0138】次に、記憶効率の評価手順（ステップS311）について説明する。図22は、記憶効率の評価手順を表わすフローチャートである。

【0139】ステップS501で記憶効率の評価を開始する。ステップS502では、主記憶に設定された未使用セクタカウンタの値と使用済セクタカウンタの値を比較する。ここで、使用済セクタカウンタの値が未使用セクタカウンタの値に対して同じか上回った場合、上位プログラムに対して記憶効率の悪化をレポートする様に構成している（ステップS502、S504）。また、未使用セクタカウンタの値が使用済セクタカウンタの値よりも大きければ、評価結果を正常とし、正常復帰する（ステップS503）。

【0140】次に、フラッシュROMの記憶領域の獲得手順（ステップS305）について説明する。図23はフラッシュROMの記憶領域の獲得手順を表わすフローチャートである。

【0141】ステップS601でフラッシュROMの記憶領域の獲得を開始する。ステップS602で未使用セクタの検索権利を獲得する。ここでは、オペレーティングシステムの提供するセマフォの機能を使用して未使用セクタの検索権利を管理している。ここでは、ステップS602からステップS609／ステップS611までの点線で囲まれた処理期間だけ未使用セクタの検索権利を独占出来る。複数のタスクが同時に同一領域を獲得する様な事態を防ぐ為のしくみである。

【0142】ステップS603でフラッシュROMの最初のセクタへポインタを移動する。ステップS603でそのセクタの管理フラグ（使用中フラグ、使用済みフラグ）を参照して、当該セクタの使用状態を判断する。使用済みか使用中ならステップS605へ分岐する。ステ

ップS605で現在ポイントしているセクタが最後のセクタならステップS611へ分岐する。この場合、使用可能な領域がフラッシュROM15に存在しないことになるので、ステップS611で未使用セクタの検索権利を開放した後、ステップS612で異常復帰する。また、ステップS605で現在ポイントしているセクタが最後のセクタでなければ、ステップS606へ分岐する。ステップS606では、ポインタを次のセクタへ移動させてからステップS604へ戻る。

【0143】ステップS604ポインタの示すセクタの管理フラグが未使用となっていればステップS607へ分岐する。ステップS607では、フラッシュROMの管理フラグを“使用中”へ変更する（使用中フラグをTRUEにする）。そして、ステップS608で、主記憶に設けた未使用セクタカウンタの値を1つ減少させる。この場合は、フラッシュROMへの記憶領域の獲得に成功しているので、ステップS609で未使用セクタの検索権利を開放し、ステップS610で正常復帰する。

【0144】次に、ガベージコレクション（ステップS312）の手順について説明する。図24はガベージコレクションの手順を表わすフローチャートである。

【0145】ステップS701でガベージコレクションを開始する。ステップS702では、整理対象のイレースブロック（以後、整理対象ブロック）を選出する。整理対象ブロックの選出手順については、図25のフローチャートを用いて詳しく説明を加える。ステップS703では、整理対象ブロックの未使用セクタを使用済化する。この使用済化の手順については、図26のフローチャートを用いて詳しく説明を加える。ここで、最初に整理対象ブロック内の未使用セクタを使用済化させる目的は、ガベージコレクション中であっても、他のタスクが整理対象ブロック内のセクタを含むセクタへの読み書きが可能な構成となっており、ガベージコレクション中に他のタスクによって整理対象ブロック内のセクタへ新たなデータが書き込まれることを防止する為である。

【0146】ステップS704では、整理対象ブロック中の使用中セクタを他の記憶領域（即ち、他のイレースブロック）へ移動させる。使用中セクタを他の記憶領域へ移動させる処理については、図27のフローチャートを参照して詳しく説明を加える。

【0147】続くステップS705では、使用中セクタの移動を終了した整理対象ブロックの消去を実行する。整理対象ブロックを消去する手順については、図28のフローチャートを参照して詳しく説明を加える。なお、この整理対象ブロックの消去において、消去回数カウンタ152の内容を主記憶にコピーしておく。ステップS705における整理対象ブロックの消去を終えると、ステップS706で主記憶に退避したデータをフラッシュROMの当該イレースブロックの消去回数カウンタへ戻す。そして、ステップS707でガベージコレクション

から復帰する。

【0148】次に、ガベージコレクションにおける整理対象ブロックの選出手順（ステップS702）について説明する。図25は整理対象ブロック選出する手順を表わすフローチャートである。

【0149】まず、ステップS801で整理対象ブロックの選出を開始する。ステップS802で評価ポイントに最初のイレースブロックをセットする。同様に、ステップS803で、整理対象候補ポイントを最初のイレースブロックにセットする。

【0150】次に、ステップS804で、評価ポイントの示すイレースブロックに使用済セクタが含まれているかどうかを判断する。使用済セクタが含まれていなければステップS804、ステップS805をスキップしてステップS807へ制御を移す。

【0151】一方、ステップS804で評価ポイントの示すイレースブロックに使用済セクタが含まれている場合には、ステップS805へ制御を移す。ステップS805では、整理対象候補ポイントの示すイレースブロックの消去回数カウンタの値と評価ポイントの示すイレースブロックの消去回数カウンタの値を比較する。もし評価ポイントの示すイレースブロックの消去回数の方が少なければステップS806へ制御を移す。ステップS806では、整理対象候補ポイントへ評価ポイントを代入する。一方、ステップS805でもし評価ポイントの示すイレースブロックの消去回数のほうが多ければそのままステップS807へ制御を移す。

【0152】ステップS807で評価ポイントが最後のイレースブロックを示しているかどうかを判断する。もし最後のイレースブロックでなければ、ステップS808で評価ポイントを次のイレースブロックへ移動させた後、ステップS804へ戻る。以上のように、ステップS804～S808の処理を繰り返すことで、整理対象候補ポイントは、使用済みセクタを含み、消去回数の少ないイレースブロックを示すようになる。

【0153】ステップS807で評価ポイントが最後のイレースブロックを示している場合はステップS809へ分岐する。ステップS809ではガベージコレクション処理（図24の処理）に復帰する。この時点の整理対象候補ポイントの示すイレースブロックが整理対象として選出される。

【0154】次に、選択された整理対象ブロック内の未使用セクタを使用済み化する処理（ステップS703）について説明する。図26は、整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【0155】ステップS901で処理を開始する。ステップS902で、整理対象ブロックの最初のセクタへポイントを移動させる。次に、ステップS903で、未使用セクタの検索権利を獲得する。これは、図23のフロ

ーチャートのステップS602と同様の効果があり、ステップS908までの点線で囲まれた間、未使用セクタの検索権利を独占する。即ち、整理対象ブロックの全セクタを対象にスキャンして未使用セクタを使用済セクタへ変更するまでの間、他のタスクが未使用セクタの検索をすることを禁止する。しかし、管理フラグのみの操作で未使用セクタを使用済セクタへ変更するので、検索権利の独占時間は短く、全体のスループットが低下することはない。

10 【0156】ステップS904で、現在のポイントが示すセクタが未使用セクタかどうかを判断する。もし未使用セクタならステップS905へ分岐する。ステップS905でその記憶を廃棄する。ステップS905の処理手順は図21のフローチャートで説明した通りである。この処理により、未使用セクタが使用済みセクタに変更される。ステップS906では、ポイントが整理対象ブロックの最後のセクタを示しているかどうかを判断する。最後のセクタを示していればステップS908へ、そうでないならステップS907へ分岐する。ステップS907ではポイントを次のセクタへ移動させてステップS904へ制御を戻す。

【0157】また、ステップS906でポイントが整理対象ブロック最後のセクタならステップS908で未使用セクタの検索権利を開放し、ステップS909でガベージコレクション処理（図24のフローチャート）へ復帰する。

【0158】次に、整理対象ブロックの使用セクタを他のイレースブロックの未使用セクタへ移動する処理（ステップS704）について説明する。図27は、整理対象ブロックの使用セクタの移動手順を表わすフローチャートである。

【0159】ステップS1000で処理を開始する。ステップS1001で整理対象ブロックの最初のセクタへポイントを移動させる。以下のステップS1002～S1012では、ポイントが指し示すセクタについて処理を行う。

【0160】ステップS1002で当該セクタの管理フラグ（使用中フラグ、使用済みフラグ）を判断する。ステップS1002で管理フラグの値が「使用中」となっていたらステップS1003へ制御を移し、「使用済」となっていたらステップS1012へ制御を移す。ステップS1003で、論理セクタをロックする。ロックしたセクタはステップS1011でアンロックされるまでの間、自タスクで占有される。

【0161】ステップS1004では記憶領域を獲得する。ステップS1004における記憶領域の確保の手順は、図23のフローチャートで説明した通りである。ここで、整理対象ブロック内の各セクタは上記ステップS703の処理で、全て使用済み化されているので、確保される記憶領域は整理対象ブロック以外のイレースブ

20

30

40

50

ックとなる。

【0162】記憶領域の獲得に成功すると、処理はステップS1008へ進む。ステップS1008では、獲得した領域へ当該セクタのデータをコピーする。そして、セクタの移動に従って、ステップS1009で記憶場所管理テーブル140を更新する。

【0163】一方、ステップS1004で記憶領域の獲得に失敗した場合は、ステップS1005へ分岐する。ステップS1005では、データ退避用の記憶領域を主記憶（DRAM）より獲得する。データ退避用記憶領域の獲得は図19のフローチャートのステップS306で説明した通りである。ステップS1006では、獲得した領域へ当該セクタのデータをコピーする。そして、ステップS1007で記憶管理テーブルを更新する。ステップS1010で元の記憶を廃棄する。即ち、ポインタの指し示すセクタの使用済みフラグをTRUEにセットする。そして、ステップS1011で当該論理セクタをアンロックする。

【0164】ステップS1012で、ポインタの指し示すセクタが、整理対象ブロックの最後のセクタかどうかを判断する。最後のセクタであればステップS1014へ、最後のセクタでなければステップS1013へそれぞれ分岐する。ステップS1013では、ポインタを次のセクタへ移動させて、ステップS1002へ戻り、次のセクタについて上述の処理を繰り返す。また、ステップS1014では、整理対象ブロック内の全てのセクタについて処理を終えているので、ガベージコレクション処理（図24のフローチャート）へ復帰する。

【0165】次に、整理対象ブロックの消去処理（ステップS705）について説明する。図28は、整理対象となったイレースブロックの消去手順を表わすフローチャートである。

【0166】ステップS1101で処理を開始する。ステップS1102で整理対象ブロックの消去回数カウンタを主記憶へコピーする。ステップS1103では整理対象ブロックの消去を実行する。ステップS1104では、主記憶へコピーした消去回数カウンタの値を1増加させた値をフラッシュROMへ書き込む。即ち、当該整理対象ブロックの消去カウンタの値を、消去処理前の値より1増加させる。その後、ステップS1105でガベージコレクション処理（図24のフローチャート）へ復帰する。

【0167】上記図24で示されるガベージコレクション処理は、極力フラッシュROMを用いた処理であり、退避データの安全性が高い。しかしながら、上述の＜未使用セクタが無い場合＞の項で説明したように、積極的に主記憶（DRAM14）を用いて使用中セクタのデータを待避し、複数個のイレースブロックを消去すると消去処理の効率がよい。但し、DRAM14にデータを待避するので、待避中のデータに関して安全性が低下する

（例えば電池が外れて電源供給が停止するとDRAMに待避したデータが失われることになる）。そこで、電源の種別を判断し、供給電源が電池の場合は待避データの安全性を重視し、ACアダプタの場合は電源供給が停止する危険性が少ないので消去処理の効率を重視するように構成してもよい。この場合の処理について図36を参照して説明する。

【0168】図36は、電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。同図において、図24のフローチャートで示される処理と同じ処理を行うステップについては同一のステップ番号を付し、ここでは詳細な説明を省略する。

【0169】ステップS1300においてガベージコレクション処理が起動されると、ステップS1301へ進み、当該装置への電源供給の形態を判断する。ここでは、図1の電源コントローラ9が、電源の供給元が電池7であるかACアダプタ23であるかを判断し、CPU5に通知する。電源種別が電池7であった場合は、ステップS1304へ進み、上述の図24で示したガベージコレクション処理を実行する。

【0170】一方、ステップS1301において電源種別がACアダプタであった場合は、ステップS1302へ進む。ステップS1302では、図24のステップS702、S703、S704に相当する処理を実行し、選出した整理対象ブロック内の未使用セクタの使用済み化と使用中セクタの待避を行う。そして、ステップS1303において、主記憶（DRAM14）にセクタの待避を行うのに十分な空き領域があるか否かを判断し、十分な空き領域があればステップS1302へ戻る。ステップS1302では、前回の整理対象ブロックとは別の整理対象ブロックを選出して、上述の処理を繰り返す。

【0171】DRAM14上に十分な空き領域が無くなると、ステップS1303からステップS1304へ進み、上述の処理で選出された整理対象ブロックの消去を行う。そして、ステップS706で主記憶に待避したデータをフラッシュROM15に戻して本処理を終了する。

【0172】以上のように、図36の処理によれば、電源がACアダプタによって供給される場合は、主記憶の空き容量を積極的に利用してデータの待避を行い、複数の整理対象ブロックを選出して、一括して消去処理を行うことができ、消去処理の効率が向上する。

【0173】なお、上記の処理では、電源種別に基づいて自動的にガベージコレクションの形態を切り換えるが、コントロールパネル12の操作により、マニュアルで切り換えるようにすることもできることはいうまでもない。

【0174】次に、基本サービスの一つである論理セクタの解放手順について説明する。図29は、論理セクタ

の解放手順を表すフローチャートである。

【 0 1 7 5 】ステップ S 1 2 0 1 で N セクタの解放を開始する。ステップ S 1 2 0 2 で N セクタをロックする。この結果、ステップ S 1 2 0 5 でアンロックされるまでの間、自タスクで論理セクタを占有出来る。続いて、ステップ S 1 2 0 3 で当該セクタの記憶を廃棄する。この記憶の廃棄処理については、図 2 1 のフローチャートで説明した通りである。ステップ S 1 2 0 4 では、D R A M 1 4 の記憶場所管理テーブル 1 4 0 へ “不在” 値を代入する。ステップ S 1 2 0 5 では、論理セクタをアンロ
10 ックし、ステップ S 1 2 0 6 で復帰する。

【 0 1 7 6 】例えば M S - D O S (商 標) 等の一般のファイルシステムでは、ファイルの消去に際しては、当該ファイルに属するセクタを F A T において上書き可能とするのみで、各セクタを解放するということを行われない。よって、このようなファイルシステムに本実施形態
20 のフラッシュ R O M 管理システムを適用すると、ファイルシステム上では無効となったデータが、有効なセクタとして残されてしまうことになり、ガベージコレクション等の効率を低下させることになる。よって、ファイルシステムの指示 (例えばファイル消去) に基づいて不要となったセクタを検出し、これを解放するように構成すれば、ガベージコレクションの効率をより向上させることができる。

【 0 1 7 7 】図 3 7 は、ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。同図において、ステップ S 1 4 0 1 でファイルシステムよりファイル消去の指示があったか否かを判断する。ファイル消去の指示があった場合は、ステップ S 1 4 0 2 へ進み、消去すべく指示された
30 ファイルに含まれるセクタを抽出する。セクタの抽出は、F A T を参照することで抽出できる。そして、ステップ S 1 4 0 3 で、先のステップ S 1 4 0 2 で抽出された各セクタについて、上記図 2 9 のフローチャートで説明したセクタの解放処理を実行する。

【 0 1 7 8 】【実施形態 2】次に実施形態 2 について説明する。

【 0 1 7 9 】＜ディスクコントローラエミュレーション＞上述の実施形態 1 で説明したフラッシュ R O M の記憶管理システムは、上位層から見た特徴がディスク媒体と、
40 良く似ている。従って、ディスクコントローラのエミュレーション機能を備えたシステムに組み込むことで、ディスクコントローラとディスク媒体をディスクコントローラエミュレーションと本実施形態の記憶管理システム (あるいは本実施形態の記憶管理システムを組み込んだ I C カード) へ置き換えることが可能となる。近年 P C M C I A に代表される I C カードが普及しているが、I C カードへディスクコントローラエミュレーション機能と上記実施形態 1 の記憶管理システムを組込むことにより、リムーバブルな記憶媒体として利用することが可能と
50

なる。第 2 の実施形態では、実施形態 1 の記憶管理システムを I C カードへ組み込んだものについて説明する。

【 0 1 8 0 】図 3 0 は実施形態 2 における I C カードの構成を表すブロック図である。同図において、2 0 0 は I C カード全体を示す。2 0 1 はマイクロコンピュータであり、ディスクコントローラエミュレーション及び記憶管理を行う。2 0 2 は R O M であり、マイクロコンピュータ 2 0 1 のプログラムを格納する。2 0 3 は R A M であり、マイクロコンピュータ 2 0 1 の主記憶として機能する。2 0 4 はフラッシュ R O M であり、上記実施形態 1 で説明した記憶管理システムによってデータを蓄積する。即ち、フラッシュ R O M 2 0 4 は、図 4 で説明した管理領域とデータ領域とで管理される。

【 0 1 8 1 】2 0 5 はコマンド / データ ・ ラッチ部であり、ホスト装置より受信した外部バスからのコマンドとシリンダ番号等を保持する。2 0 6 は F I F O メモリであり、先入れ先出し方式でデータの入出力を行う。2 0 7 はタプル R O M であり、当該カードの特徴等を記憶しており、外部バスからのみ読み出しができる。

【 0 1 8 2 】上述の各構成の機能は、以降の動作説明でより明らかとなる。

【 0 1 8 3 】図 3 1 は、本実施形態 2 の I C カードを利用する為のホストシステムの簡単なブロック図である。同図において、3 0 1 はホストシステム側のマイクロコンピュータである。3 0 2 はカードインターフェースであり、ホストシステムの内部バスと I C カード 2 0 0 の外部バスを接続する。なお、カードインターフェース 3 0 2 は、I C カード 2 0 0 への電源供給を行うための電源供給線や、I C カード 2 0 0 からの割り込み要求 (I R Q 出力) を受け付けるための信号線も備えている。

【 0 1 8 4 】図 3 2 は、図 3 1 のホストシステムが I C カードを接続する際の手順を示すフローチャートである。ステップ S 4 1 0 0 で処理を開始すると、ステップ S 4 1 0 1 で I C カードへの電源供給を開始する。ステップ S 4 1 0 2 では、I C カード 2 0 0 内のタプル R O M 7 から、タプル形式で格納されているデータを解析する。タプル R O M 7 の内容を解析することで、接続されている I C カードの特徴が分かる。

【 0 1 8 5 】ステップ S 4 1 0 3 では、ステップ S 4 1 0 2 で解析したタプル情報によって、接続されている I C カードが内部バスへ接続可能かどうかを判断する。そして、接続可能ならステップ S 4 1 0 4 へ、接続不可能ならステップ S 4 1 0 5 へとそれぞれ分岐する。ステップ S 4 1 0 4 では、I C カード側のバスをホストの内蔵バスのメモリ空間と I O 空間へマッピングする。この時点でホスト装置のバスの空間にディスクコントローラが有るのと同じ状態になる。

【 0 1 8 6 】図 3 3 は I C カード 2 0 0 内のマイクロコンピュータ 1 のメインシーケンスを示すフローチャートである。ステップ S 4 2 0 1 で I C カードの電源が投入

されると、ステップ S 4 2 0 2 で記憶管理システムの初期化を行う。即ち、フラッシュ ROM 2 0 4 の全イレースブロックの論理セクタの状態を一旦読み出し、読み出した情報に従って主記憶用の RAM 2 0 3 へ記憶場所管理テーブルを作成する。ステップ S 4 2 0 3 で主記憶上のコマンドバッファとしてリング状のバッファを用意して初期化し、割り込み処理を許可する。この処理以降割り込みルーチンの動作が始まる。

【0187】割り込みルーチンのシーケンスを図 3 4 のフローチャートに示す。割り込みルーチンの動作を理解した方が、図 3 3 のフローチャートの説明が容易となる為、ここで図 3 4 のフローチャートについて説明を行う。

【0188】ホストシステムがコマンド/データ・ラッチ 2 0 5 へのコマンドのアドレスへコマンドを書き込むと、コマンド/データ・ラッチ 2 0 5 からマイクロコンピュータ 2 0 1 へ割り込みが発生する。コマンド/データ・ラッチ 2 0 5 は、ホストバスと IC カード内部のバスの IO アドレス空間にマッピングされていて、コマンド/データはそれぞれ図 3 5 に示すように IO アドレスが割り振られている。図 3 5 は、本実施形態のコマンド/データ・ラッチにおける IO 割り付けを示す図である。本例では、図 3 5 中の Command のアドレスにコマンド（例えばデータの読出しを指示する ReadSector(s)）を書き込むことでマイクロコンピュータ 2 0 1 へ割り込みが発生する。

【0189】割り込みが発生すると、マイクロコンピュータ 2 0 1 のソフトウェアは、図 3 4 のフローチャートのステップ S 4 3 0 1 へ制御を移す。ステップ S 4 3 0 2 では、コマンド/データ・ラッチ 2 0 5 に書き込まれたデータを読み出して、主記憶上のリングバッファへデータを格納する。ステップ S 4 3 0 3 で割り込みルーチンを終了して図 3 3 のフローチャートへ復帰する。

【0190】図 3 3 のフローチャートの説明に戻る。ステップ S 4 2 0 4 でマイクロコンピュータ 2 0 1 はコマンドバッファの状態を判断する。コマンドバッファへデータが格納されていれば、ステップ S 4 2 0 5 へ分岐し、データが格納されていなければステップ S 4 2 1 3 へ分岐する。ステップ S 4 2 1 3 では CPU を休止状態にする。多くのワンチップマイクロコンピュータは、命令の実行を休止して消費電流を減らす機能を備えているが、本実施形態の CPU もこの種の機能を備える。そして、IRQ による割り込み要求信号が入力されると、CPU 2 0 1 は休止状態から復帰して上述の割り込みルーチンを実行する。割り込みプログラムの実行が済んだ時点でステップ S 4 2 1 3 から復帰してステップ S 4 2 0 4 へ戻る。

【0191】ステップ S 4 2 0 4 でコマンドバッファへデータが格納されていると、ステップ S 4 2 0 5 へ移行する。ステップ S 4 2 0 6 では、リングバッファからデ

ータを読み出す。ステップ S 4 2 0 6 でコマンドを解釈する。Seek コマンドの場合はステップ S 4 2 0 7、Read Sector(s) コマンドの場合はステップ S 4 2 0 8 へ、Write Sector(s) の場合はステップ S 4 2 0 9 へ、Identify Drive コマンドの場合はステップ S 4 2 1 0 へそれぞれ分岐する。他にもコマンドがあるが本実施形態の説明上重要でないものは省き、フローチャートを簡略化している。ステップ S 4 2 0 7 ~ 4 2 1 0 までのコマンドの実行を終了したらステップ S 4 2 0 4 まで戻り、上記の処理を繰り返す。

【0192】ステップ S 4 2 0 7 では、Seek コマンドを実行する。Seek といってもフラッシュ ROM には、ディスクデバイスと違ってヘッドが無いので、次のコマンドに備えての妥当性等をチェックするだけである。IC カードのサポートするヘッド数を超えるヘッド位置などを指定された場合は、ディスク装置同様にエラーが発生する。

【0193】ステップ S 4 2 0 8 は ReadSector(s) コマンドに対する処理を行う。ReadSector(s) コマンドは、読み出すべきセクタの個数が図 3 5 の SectorCount で指定される。よって、ステップ S 4 2 0 8 では、指定された場所のセクタを SectorCount 個読み出す行を行う。本実施形態の記憶管理システムでは、リニアな論理セクタ番号を使って管理を行っているので、シリンダ/ヘッド/セクタ番号を元にリニアな論理セクタ番号を計算し、論理セクタの内容を FIFO メモリ 2 0 6 へ転送し、コマンド/データ・ラッチ 2 0 5 の SectorNumber のインクリメントも行う。FIFO メモリ 2 0 6 は、IC カード 2 0 0 の内部バスから書き込んだデータを外部バスから読み出すことができ、また外部バスから書き込んだデータを IC カード内部バスから読み出す構成となった FIFO メモリである。

【0194】ここで、上述のリニアな論理セクタ番号について説明する。一般にハードディスクに対して指定する番号は、セクタ、シリンダ、ヘッドのパラメータで決まる 3 次元の不連続な番号である。例えば、シリンダ数が 1 0 2 4 個、ヘッド数が 1 6 個、セクタ数が 6 3 個のハードディスクの場合、セクタ数は $1 0 2 4 \times 1 6 \times 6 3 = 1 0 3 2 1 9 2$ 個となる。

【0195】このセクタを 0 番から 1 0 3 2 1 9 2 番としてアクセスできると良いのであるが、上記の 3 つのパラメータをすべて指定してアクセスするように設計されている。例えば、シリンダ 5 0 0 ・ヘッド 1 6 ・セクタ 6 3 の次は、シリンダ 5 0 1 ・ヘッド 0 ・セクタ 1 をアクセスするといった具合である。なお、これら 3 つのパラメータをそれぞれの頭文字をとって CHS パラメータと呼ぶ。

【0196】MS-DOS（商標）のようなオペレーティングシステムでは、内部ではリニア（連続的）なセクタ番号を用いるが、デバイスドライバがこれを CHS パ

10

20

30

40

50

ラメータに変換する。本実施形態のシステムでは、リニアなセクタ番号を用いるのでCHSパラメータの値を元にリニアなセクタ番号を求める。上記で挙げたハードディスクの場合は、

シリンダ番号×(16×63)+ヘッド番号×(63)+セクタ番号

を計算することで、リニアな論理セクタ番号が求まる。

【0197】ステップS4209はデータラッチで指定された場所のセクタヘデータを書き込む処理を行う。データは、FIFOメモリ206経由でホストシステムから受け取る。

【0198】ステップS4210は、ICカード200がどのようなハードディスクをエミュレーションしているかという情報を返す処理を行う。すなわちシリンダ数やModelNumberなどハードディスクとしてのスペックを含むデータをFIFOメモリ206へ書き込む処理を行う。

【0199】＜ファイルシステムの解析＞以上説明した様に実施形態1で説明した記憶管理システムをICカードに組み込むことで、ATAハードディスク等の置き換え用途に使用できる。しかし、ATAコマンド等のFATキャッシュやファイル消去によって生じた不要セクタの開放といった処理を行う為の情報を上位システムからもらう手法が無い。ATAコマンドの空き部分を利用してセクタの開放コマンドとキャッシュするセクタ番号指定コマンドを追加実装することで、FATキャッシュと不要セクタ解放の機能が実現できる。そして、この様な機能があることを想定していない現状のMS-DOS等のシステムでも、FATキャッシュやセクタの開放を実現できた方が良いことは言うまでもない。

【0200】FATシステムは論理セクタ番号0に相当する部分にFATの場所やサイズといった情報を格納している。本実施形態では、このセクタを読むことでFATの場所やサイズを取得し、FATキャッシュの処理に利用する。同様に本来書き込みデータの内容を理解しないはずのICカードであるが、ファイルシステムの為の情報(ディレクトリエントリやFAT)を解析することでICカードが自立的に不要セクタを判断して開放する等の処理に役立てることが出来る。もちろんFATに限った話では無く、HPFSやマッキントッシュ(商標)のファイルシステムでも書き込むデータの内容を解析すれば、不要セクタの検出が可能である。この様に構成することでATAハードディスクのインタフェースでも、ファイルシステムの動作に合わせた最適化処理を行うことを可能にする。

【0201】上記装置の機能もしくは方法の機能によって達成される本発明の目的は、前述の実施形態のプログラムを記憶させた記憶媒体によっても達成できる。例えば、パーソナルコンピュータに、その記憶媒体を装着し、その記憶媒体から読み出した以下に説明するような

フラッシュROM管理プログラムを実行することにより、フラッシュROMをディスクシステムと同等に使用できるようになるとともに、マルチタスクシステムに好適に対応することが可能となる。このための本発明にかかるプログラムの構造的特徴は、図49、図50に示す通りである。

【0202】図49、図50は本実施形態における記憶媒体に格納される制御プログラムの制御手順、及び本記憶媒体のメモリマップを示す図である。

【0203】図49(a)において、350は管理処理であり、データ領域とこれに対応する管理領域とでセクタを構成し、複数のセクタをフラッシュROMに形成し、各セクタにおいてデータ領域の記憶状態を示す状態情報(セクタ番号154、使用中フラグ155、使用済フラグ156)を管理領域に格納し、該状態情報に基づいてフラッシュROMのアクセスを管理する。例えば、図18や図19のフローチャートで示したように、セクタ単位でのフラッシュROMへのデータの書き込みや読み出しを制御する。

【0204】351は排他処理であり、並列に実行可能な複数のタスクの一つによって前記複数の記憶ブロックの一つがアクセスされた場合、そのアクセスされた記憶ブロックについて、他のタスクからのアクセスを禁止する。例えば、図18のステップS202、図19のステップS302における論理セクタのロック処理がこれに該当する。

【0205】上記制御手順を実現するための制御プログラムは、フロッピーディスクやハードディスク、あるいはCD-ROM等の記憶媒体に、例えば図49の(b)のメモリマップに示すような構成で格納される。上記制御プログラムは、例えばパーソナルコンピュータ等の情報処理装置によって読み出され、主記憶(RAM)上にロードされて、CPUにより実行される。なお、主記憶上への上記制御プログラムのロードは、LANを介して行われてもよい。

【0206】なお、図49(b)において、管理処理モジュール350'、排他処理モジュール351'は、それぞれ管理処理350、排他処理351を実行するプログラムモジュールである。

【0207】また、図50の(a)において、管理処理360は上記管理処理350と同様の処理を実現する。消去用前処理361は、状態情報が無効データを示す記憶ブロックに対して消去速度を向上するための前処理を行う。即ち、図38に示した前処理を実行する処理である。そして、実行処理362は、付与された優先度に従って実行がスケジューリングされる複数のタスクのうちの低優先度のタスクに前記消去用前処理の実行を割り当てて実行する。プライオリティーを設定可能なマルチタスクシステムに適用する場合に、当該消去用前処理を低優先度のタスクで実行させる。これにより、消去用前処

理が他のタスクに影響を及ぼすことが防止される。

【0208】上記図50の(a)の制御手順を実現するための制御プログラムは、図49の制御プログラムと同様に、フロッピーディスクやハードディスク、あるいはCD-ROM等の記憶媒体に、例えば図50の(b)のメモリマップに示すような構成で格納される。上記制御プログラムは、例えばパーソナルコンピュータ等の情報処理装置によって読み出され、主記憶(RAM)上にロードされて、CPUにより実行される。なお、主記憶上への上記制御プログラムのロードは、LANを介して行

われてもよい。
【0209】なお、図50(b)において、管理処理モジュール360'、消去用前処理モジュール351'、実行処理モジュール352'は、それぞれ制御手順で示した管理処理350、消去用前処理351、実行処理352の各処理を実行するプログラムモジュールである。

【0210】また、本発明は、複数の機器から構成されるシステムに適用しても、1つの機器からなる装置に適用してもよい。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。この場合、本発明に係るプログラムを格納した記憶媒体が、本発明を構成することになる。そして、該記憶媒体からそのプログラムをシステム或は装置に読み出すことによって、そのシステム或は装置が、予め定められた仕方で作動する。

【0211】

【発明の効果】以上説明したように本発明によれば、フラッシュROMをファイルシステムに適応させることが可能な管理方式を、マルチタスクシステムに好適に適用することが可能となる。

【0212】また、本発明によれば、複数のタスクが複数のフラッシュROMに対して効率良く書込みを行える。

【0213】また、本発明によれば、低優先度のタスクで消去処理の処理速度向上のための前処理を行うことが可能となり、ガベージコレクション処理等を高速に行えるようになる。

【0214】

【図面の簡単な説明】

【図1】実施形態1におけるカメラシステムの構成を表すブロック図である。

【図2】本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。

【図3】デバイスドライバの管理ブロックをC言語で記述した宣言文を示す図である。

【図4】フラッシュROM上のセクタ構造の例を示す図である。

【図5】管理領域用データと、データ領域とを分離して格納する構成を表す図である。

【図6】各フラグの状態に対応した意味を示す図であ

る。

【図7】フラッシュROMにおけるセクタの書き換え手順を説明する図である。

【図8】本実施形態におけるフラッシュROMのガベージコレクション動作を説明する図である。

【図9】未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【図10】DRAM上に作成された記憶場所管理テーブルを説明する図である。

【図11】キャッシュソフトウェアの階層的な位置付けを表す図である。

【図12】キャッシュの主記憶上のデータ構造を表わす図である。

【図13】FATキャッシュの読み出し手順を表すフローチャートである。

【図14】FATキャッシュの書込み手順を表すフローチャートである。

【図15】データ書き込み完了を確認するための動作手順をC言語で表現した図である。

【図16】DC/DCコンバータの出力容量を越えない様に電源を管理するプログラムをC言語で表現した図である。

【図17】本実施形態のリポートからサービスの開始までの動作手順を表わすフローチャートである。

【図18】指定セクタの読み出しサービスの手順を表わすフローチャートである。

【図19】論理セクタの書き込みサービスの手順を表わすフローチャートである。

【図20】主記憶上に獲得した退避データリストの様子である。

【図21】記憶を破棄する手順を表わすフローチャートである。

【図22】記憶効率の評価手順を表わすフローチャートである。

【図23】フラッシュROMの記憶領域の獲得手順を表わすフローチャートである。

【図24】ガベージコレクションの手順を表わすフローチャートである。

【図25】整理対象ブロック選出する手順を表わすフローチャートである。

【図26】整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【図27】整理対象ブロックの使用セクタの移動手順を表わすフローチャートである。

【図28】整理対象となったイレースブロックの消去手順を表わすフローチャートである。

【図29】論理セクタの解放手順を表すフローチャートである。

【図30】実施形態2におけるICカードの構成を表すブロック図である。

【図31】本実施形態2のICカードを利用する為のホストシステムの簡単なブロック図である。

【図32】図31のホストシステムがICカードを接続する際の手順を示すフローチャートである。

【図33】ICカード内のマイクロコンピュータのメインシーケンスを示すフローチャートである。

【図34】ICカード内のマイクロコンピュータの割り込み処理の手順を表すフローチャートである。

【図35】IOアドレスの割り付け状態を表す図である。

【図36】電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。

【図37】ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。

【図38】本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。

【図39】本実施形態におけるフラッシュROMへの1バイトデータの書き込み手順を表すフローチャートである。

【図40】電源の共有手順を説明するためのフローチャートである。

ートである。

【図41】本実施形態におけるフラッシュROMへの制御プログラムの格納状態を説明する図である。

【図42】相対アドレスで表現されたプログラムコードの一例を表す図である。

【図43】図42のリロケーション情報レコードのデータを格納するテーブルを表す図である。

【図44】図41のプログラムを主記憶の8710番地へマッピングした場合のプログラムコードを示す図である。

【図45】ディレクトリスロットの特徴を表す図である。

【図46】図45のディレクトリスロットにおいて、FileBが削除された状態を示す図である。

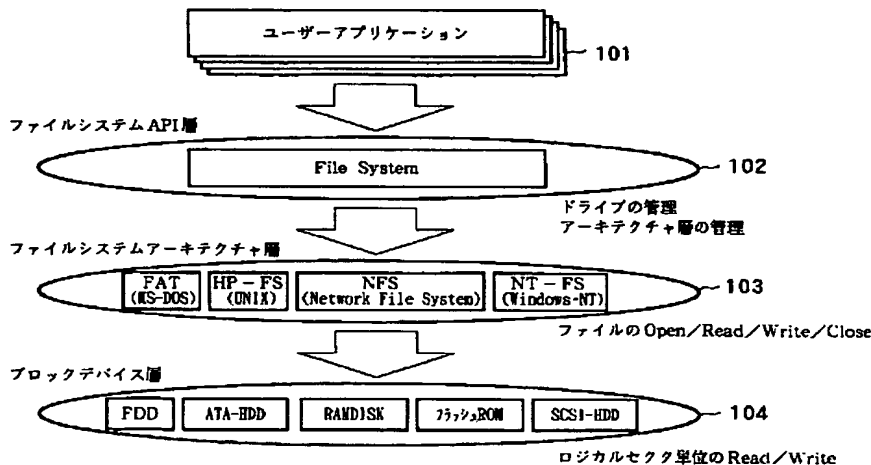
【図47】本実施形態のDOS互換ファイルシステムでファイルを消去した後の状態を表す図である。

【図48】本実施形態による電源資源（セマフォ）の分割利用を説明するフローチャートである。

【図49】本実施形態の制御を実現する制御プログラムを提供する記憶媒体の内容を説明する図である。

【図50】本実施形態の他の制御を実現する制御プログラムを提供する記憶媒体の内容を説明する図である。

【図2】



【図3】

```

1: typedef struct DeviceTag {
2:     struct DeviceTag * Next :
3:     char DevName [32] :
4:     int (* InitDev) (void) :
5:     int (* ShutDown) (void) :
6:     int (* ReadSector) (long lsect, long nsect, char * buffer) :
7:     int (* WriteSector) (long lsect, long nsect, char * buffer) :
8:     int (* ReleaseSector) (long lsect, long nsect) :
9: } Device ;

```

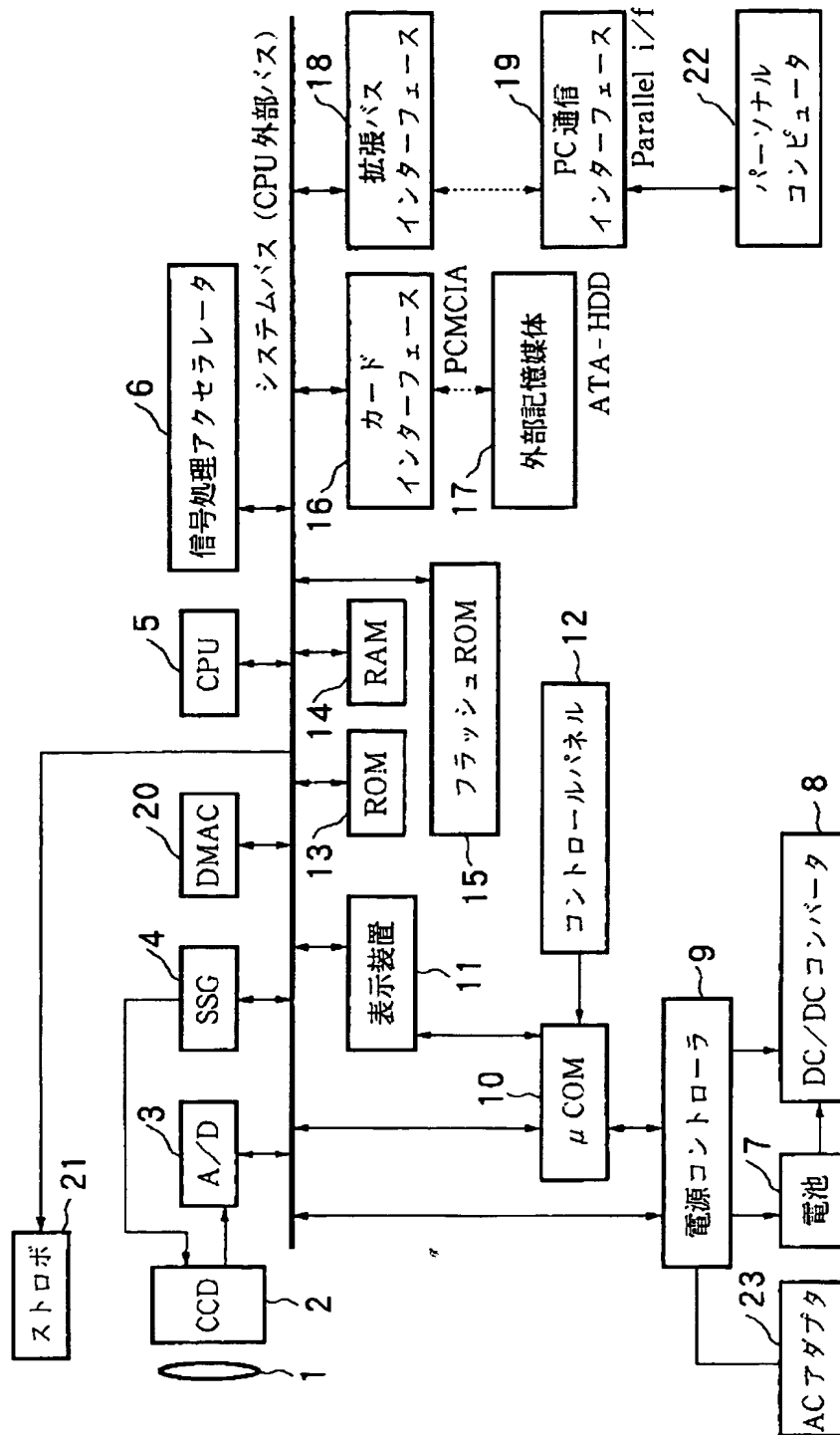
【図41】

識別コード
ID=Program Size =xxx Program
...
ID=Rlocate Size =xxx RlocateInfo
...

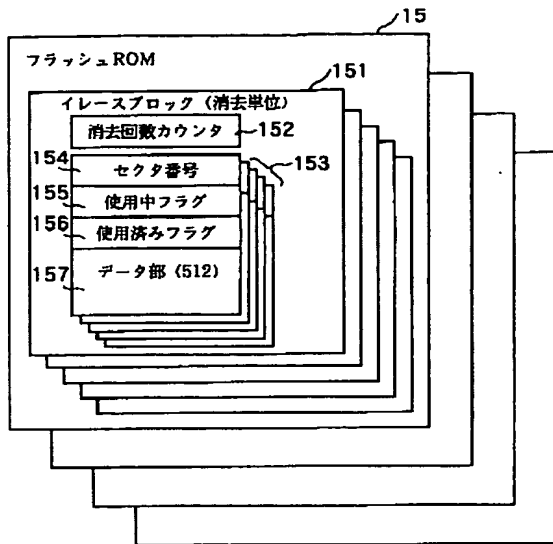
【図42】

相対アドレスで表現されたプログラムコード				
番地	データ	ニーモニック		
0040	1234	DB	1234	
0060	85,0010	JMP	0010	
...
0060	85,-20	MOV	A,@(-20)	
...

【図 1】



【図4】



【図15】

```

1: void WriteByteEEP (struct DEV * Dev,char * Address,char Data)
2: {
3:     while (* Dev -> Address != Dev -> Data) :
4:         * Address = Data :
5:         Dev -> Address = Address :
6:         Dev -> Data = Data :
7:         RotateRdyQueue0 :
8: }
9: char ReadByteEEP (struct DEV * Dev,char * Address)
10: {
11:     if (Dev -> Address == Address) return Dev -> Data :
12:     while (* Dev -> Address != Dev -> Data) :
13:         return * Address :
14: }

```

【図5】

セクタ番号テーブル

セクタ番号
セクタ番号
セクタ番号
セクタ番号
セクタ番号

フラグテーブル

フラグ
フラグ
フラグ
フラグ
フラグ

データテーブル

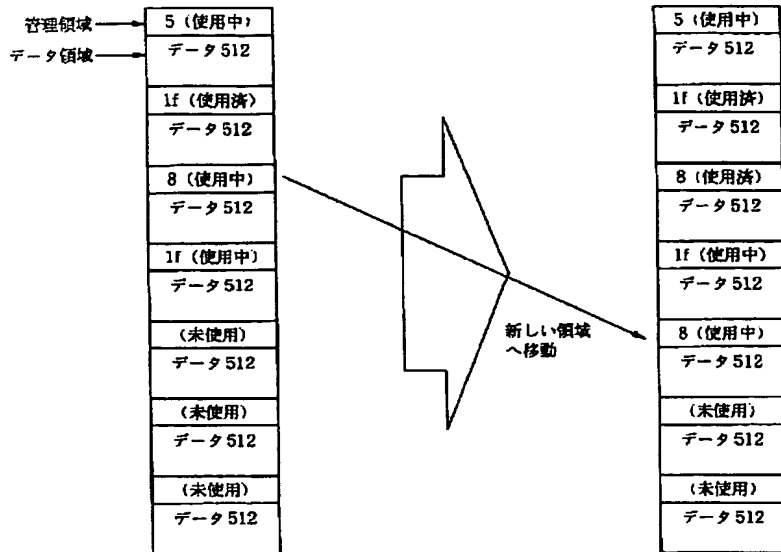
データ512
データ512
データ512
データ512
データ512

【図6】

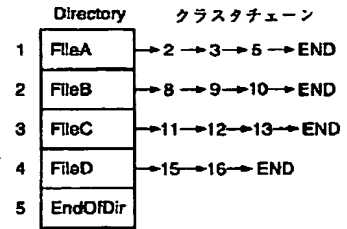
155 使用中フラグ	156 使用済みフラグ	意味
FALSE	FALSE	「未使用セクタ」... 消去後の状態
TRUE	FALSE	「使用中セクタ」... セクタ番号が示すセクタをデータ部に格納している
TRUE	TRUE	「使用済みセクタ」... データ部に格納されているデータは無効であり消去するまでこのセクタを利用できない

FALSE : 消去後の状態 TRUE : FALSE のビット反転値

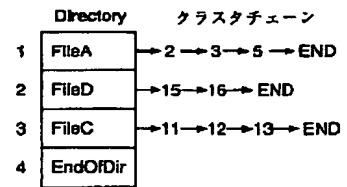
【図7】



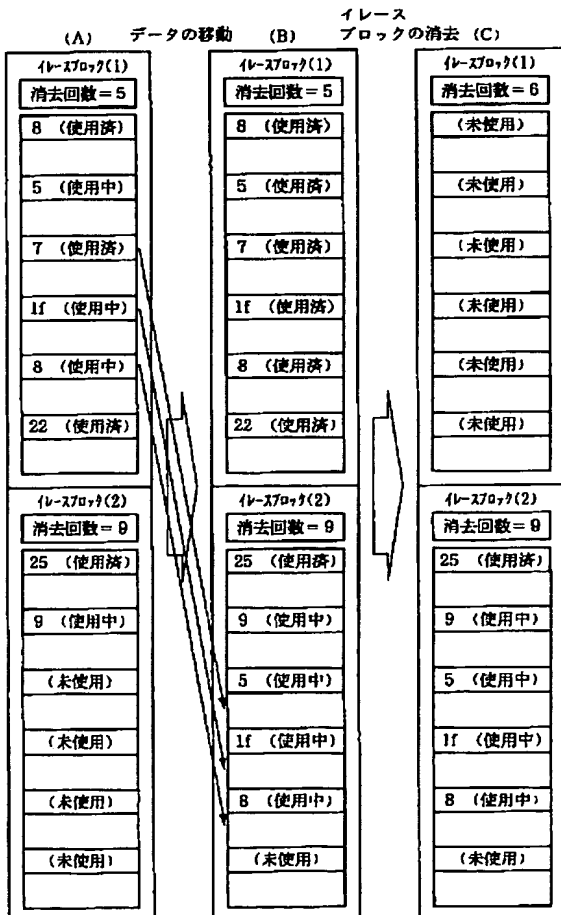
【図45】



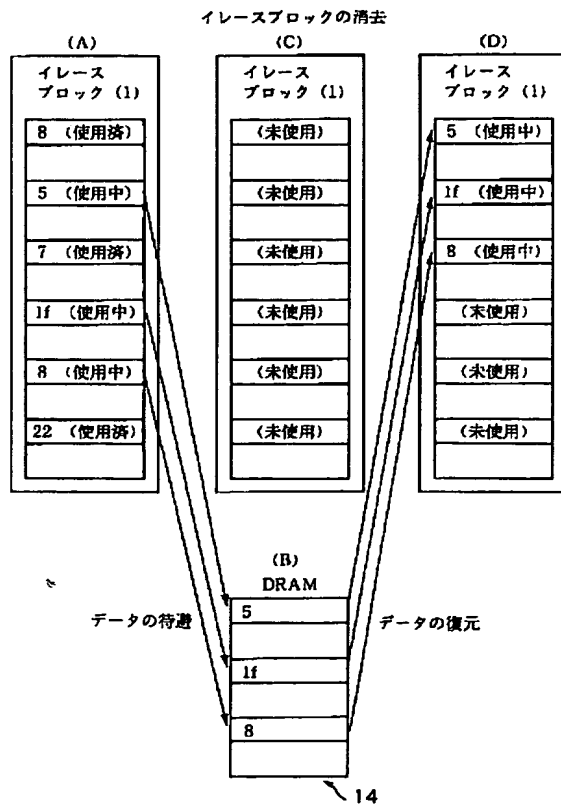
【図47】



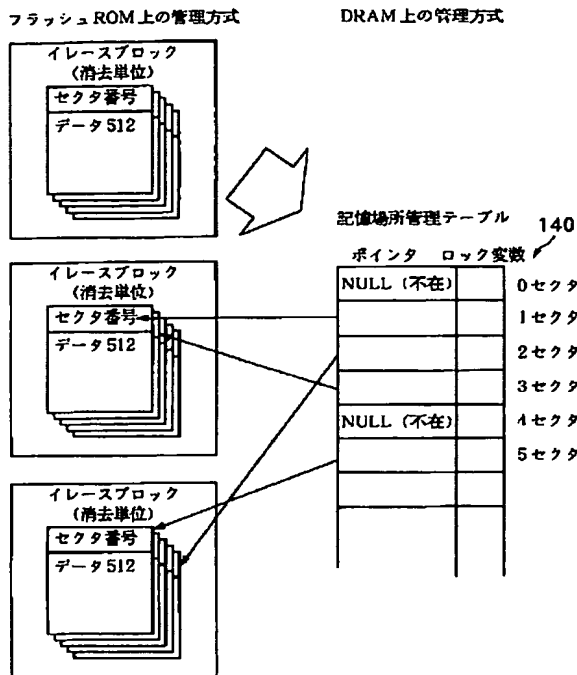
【図8】



【図9】



【図 10】



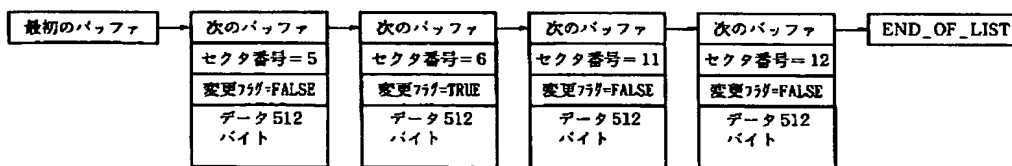
【図 16】

```

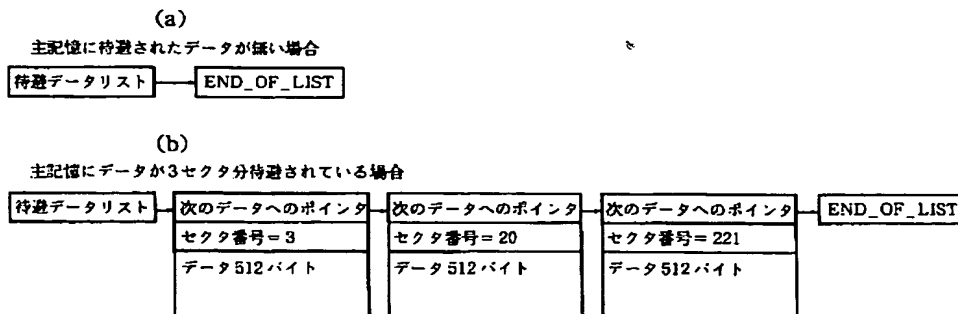
1: void ZoomUp (void)
2: {
3:     WaitSemapho (SemDCDC) :
4:     Motor1StepUp (void) :
5:     SignalSemapho (SemDCDC) :
6: }
7: char XWriteSectorEEP (struct * Dev, int Sector, char * Buffer)
8: {
9:     WaitSemapho (SemDCDC) :
10:    WriteSectorEEP (Dev, Sector, Buffer)
11:    While (* Dev -> Address != Dev -> Data) :
12:    SignalSemapho (SemDCDC) :
13: }

```

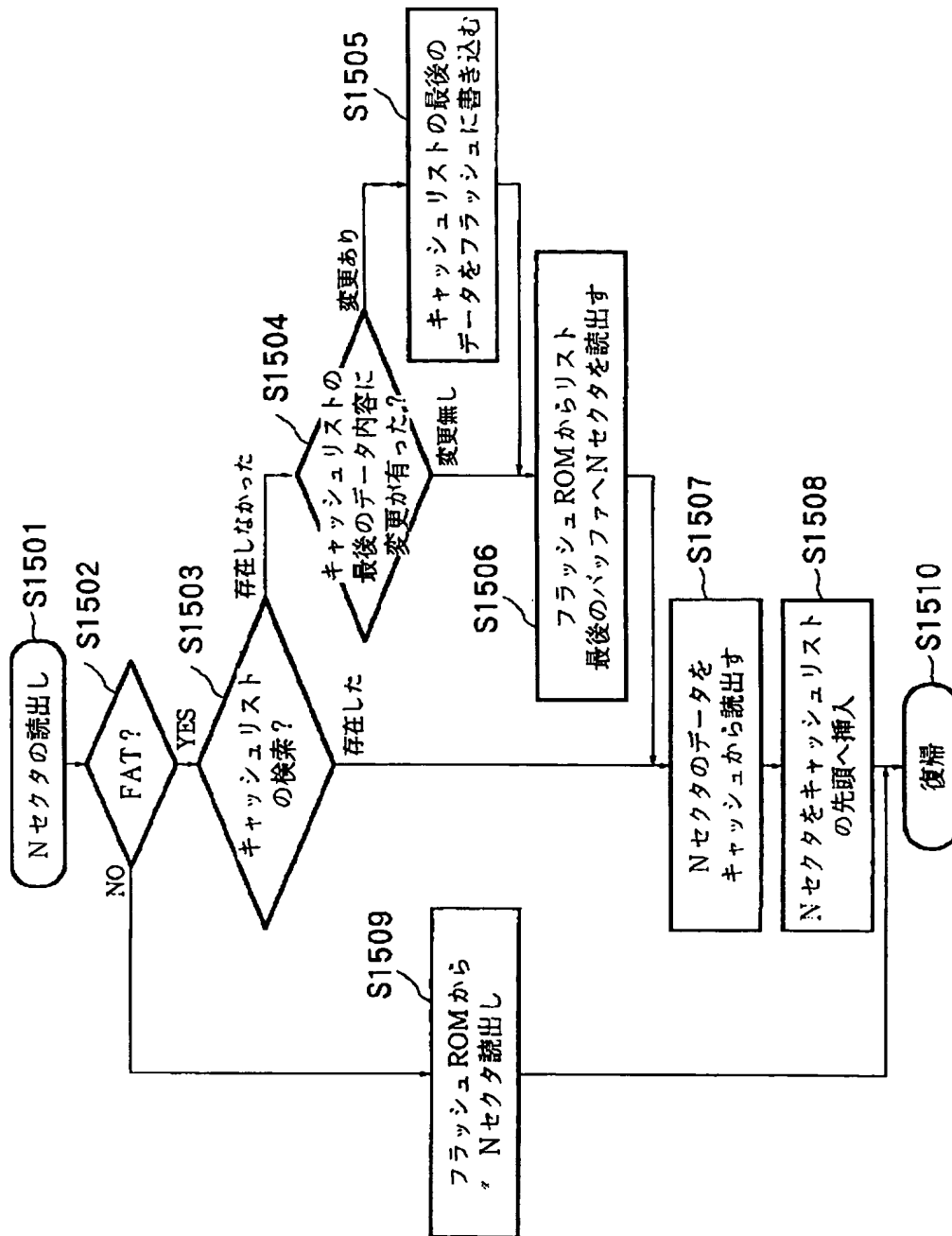
【図 12】



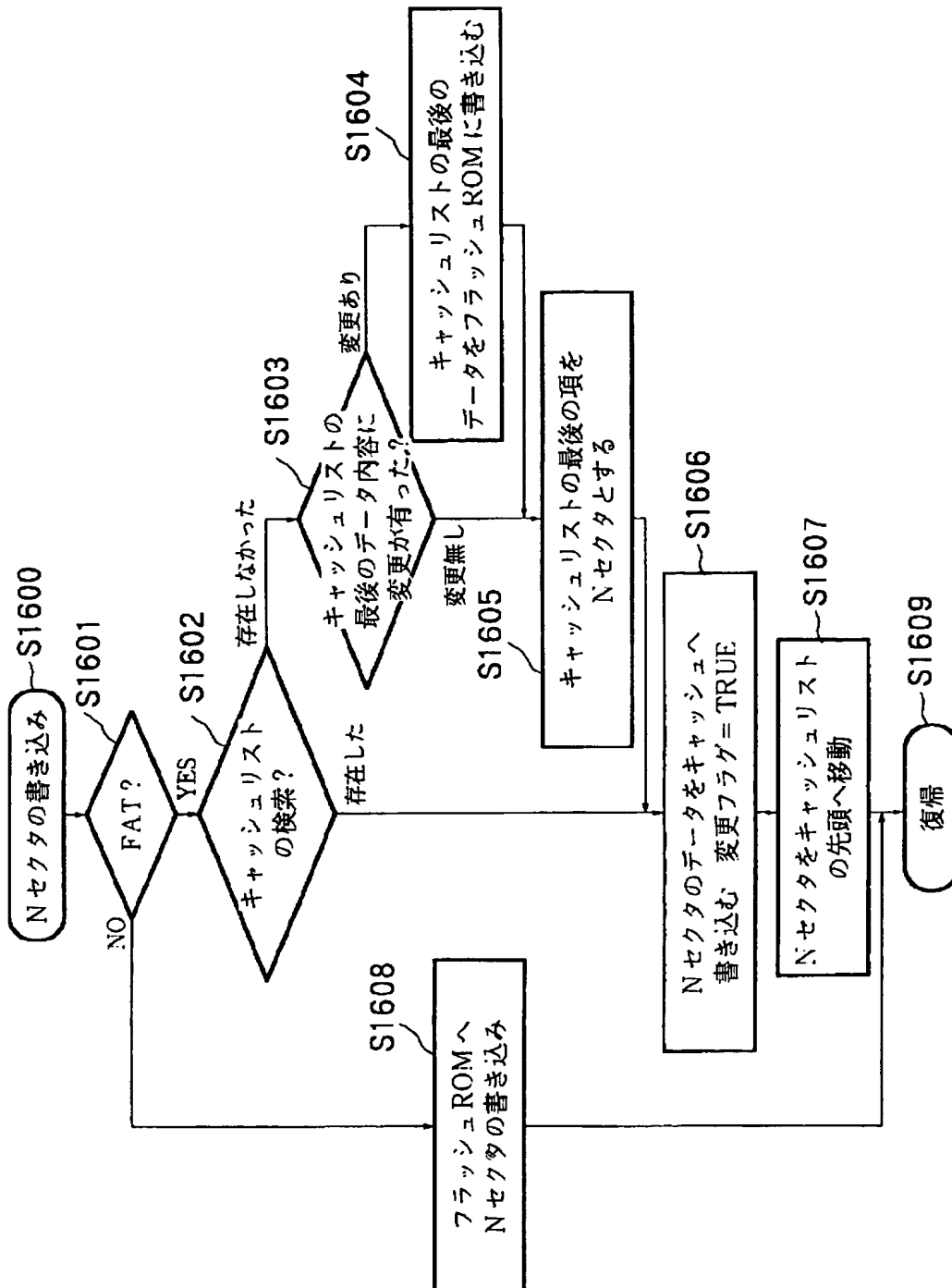
【図 20】



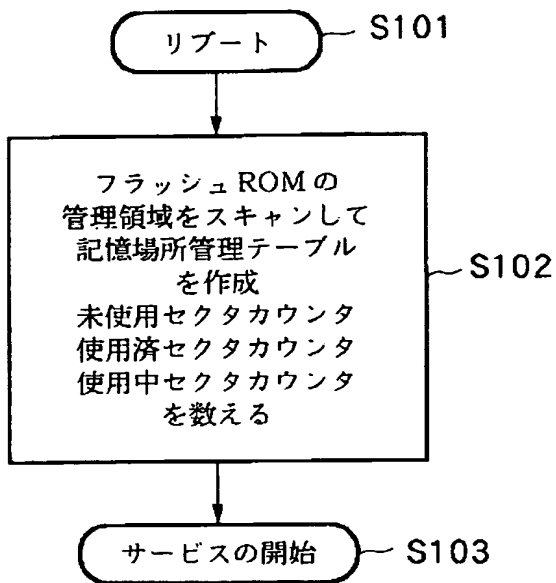
【図 13】



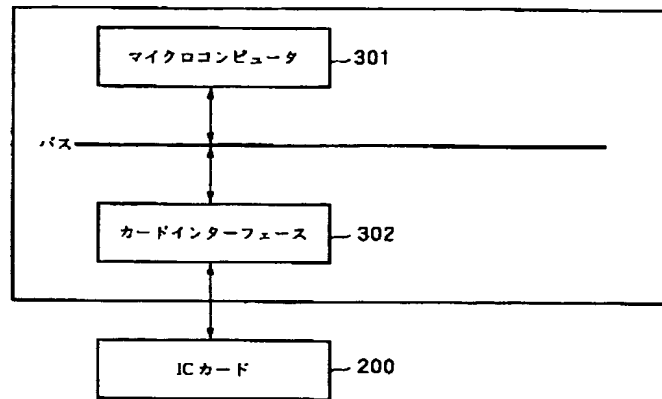
【図14】



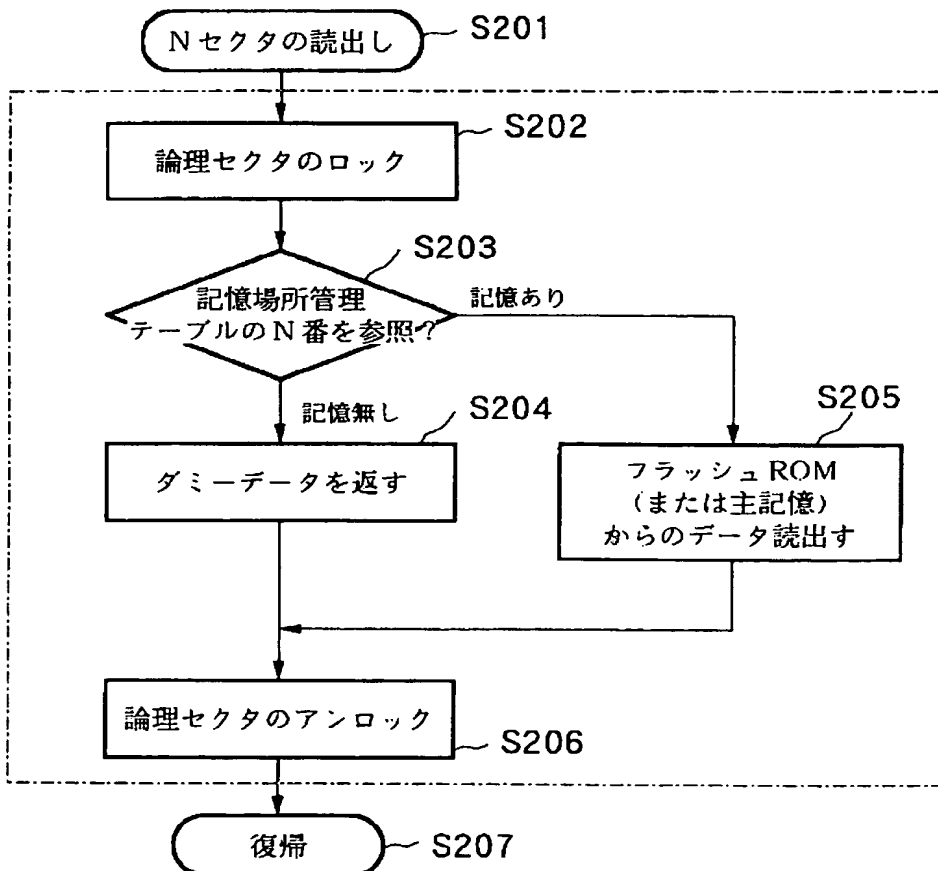
【図 17】



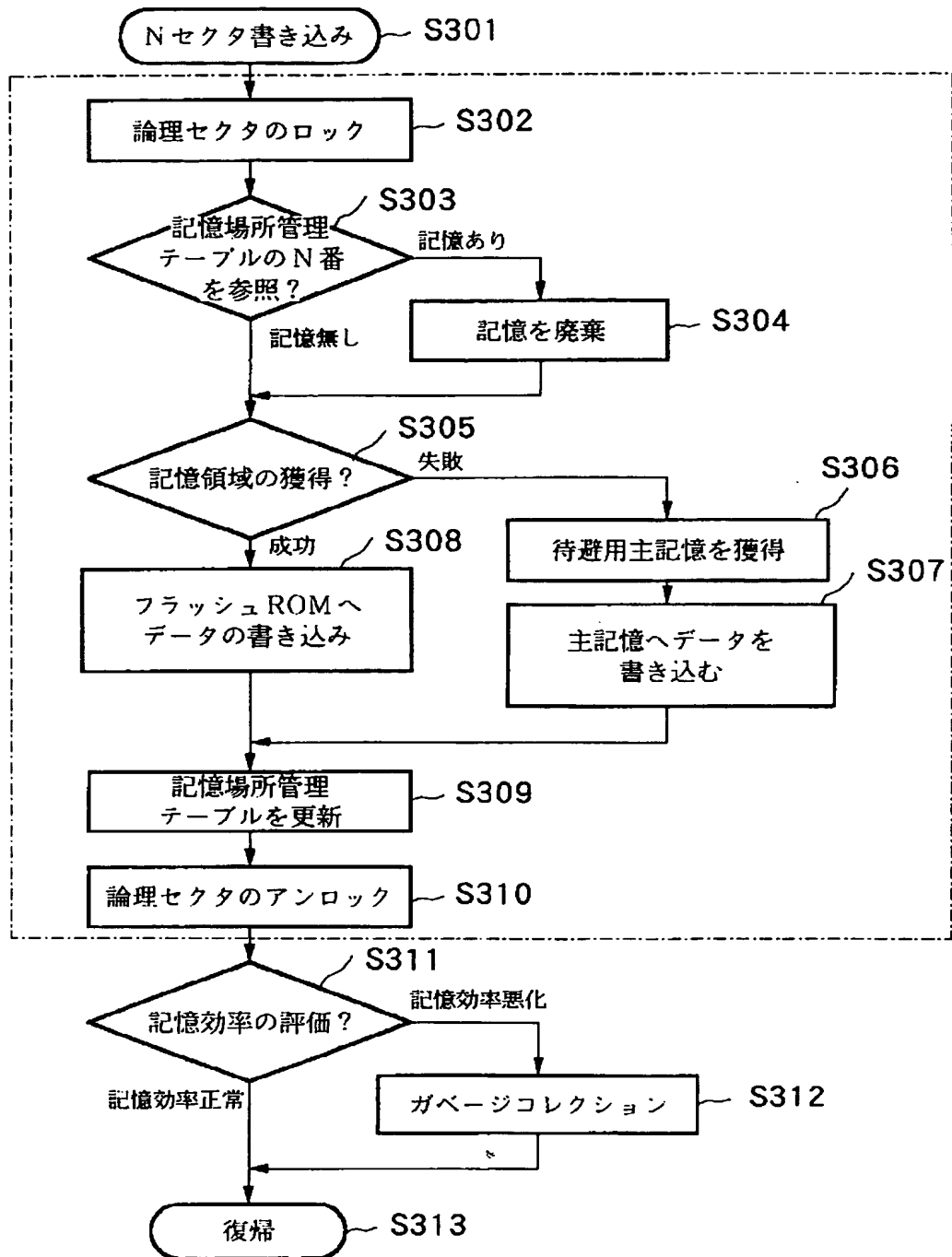
【図 31】



【図 18】



【図 19】

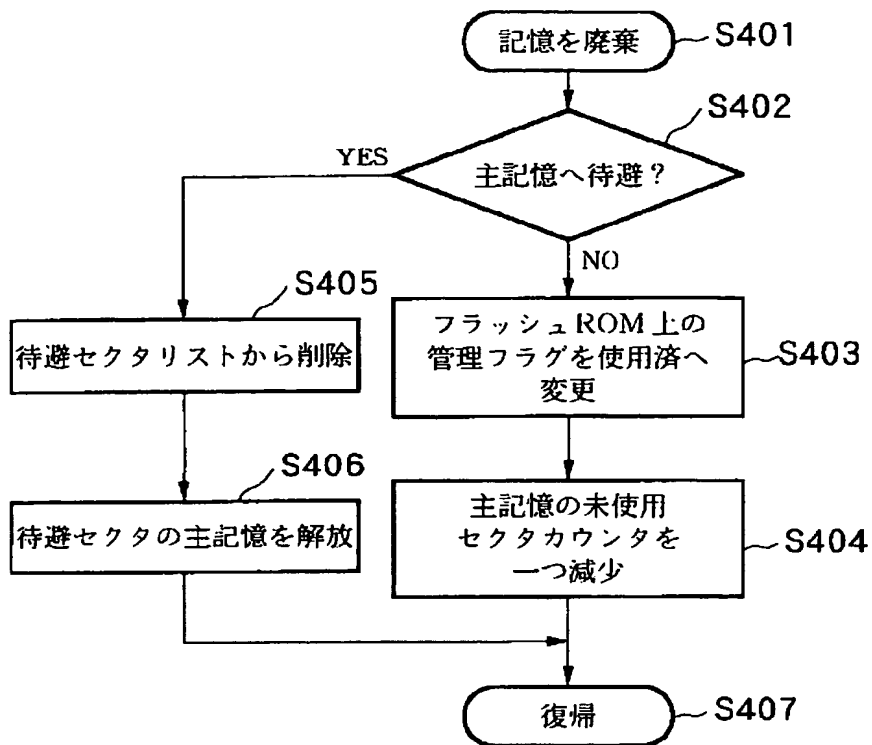


【図 43】

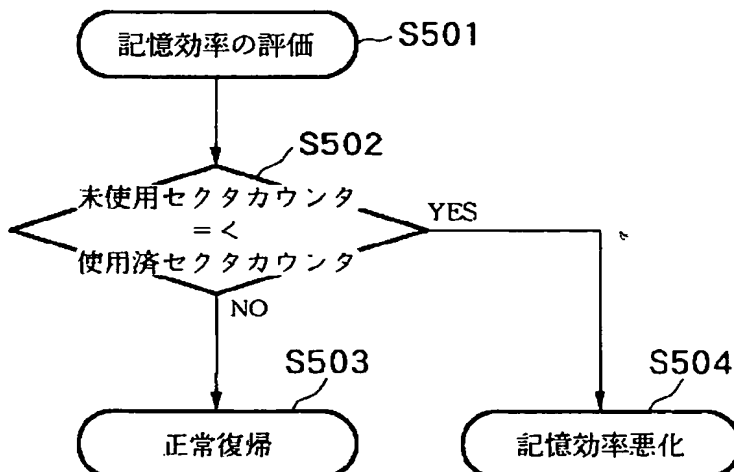
プログラム中で絶対アドレスに変換しなければならない番地のテーブル

0052
0062
:

【図 2 1】



【図 2 2】

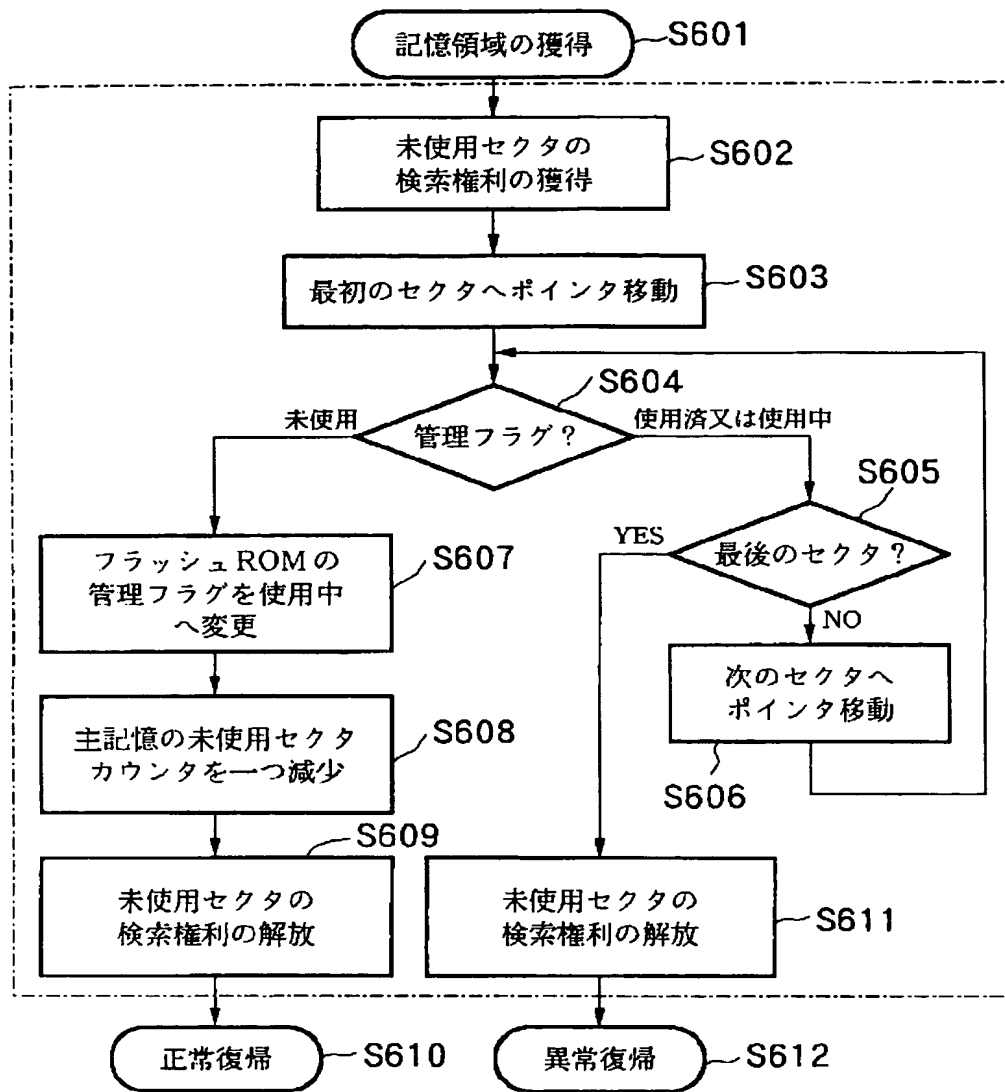


【図 4 4】

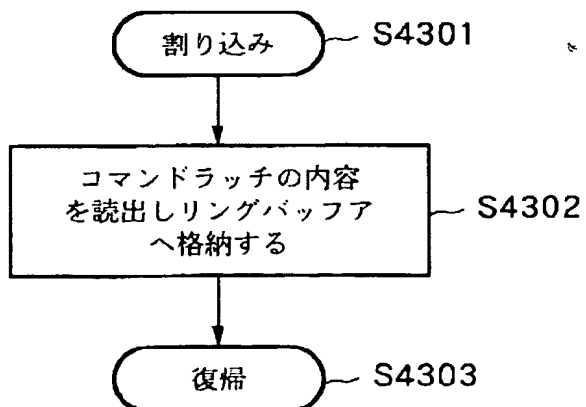
主記憶の8710番地へマッピングされたプログラムコード

番地	データ	ニーモニック
8750	1234	DB 1234
8760	85,8770	JMP 8770
...
8770	88,8750	MOV A,@(8750)
...

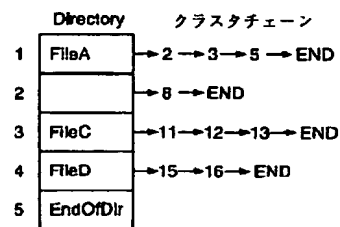
【図 2 3】



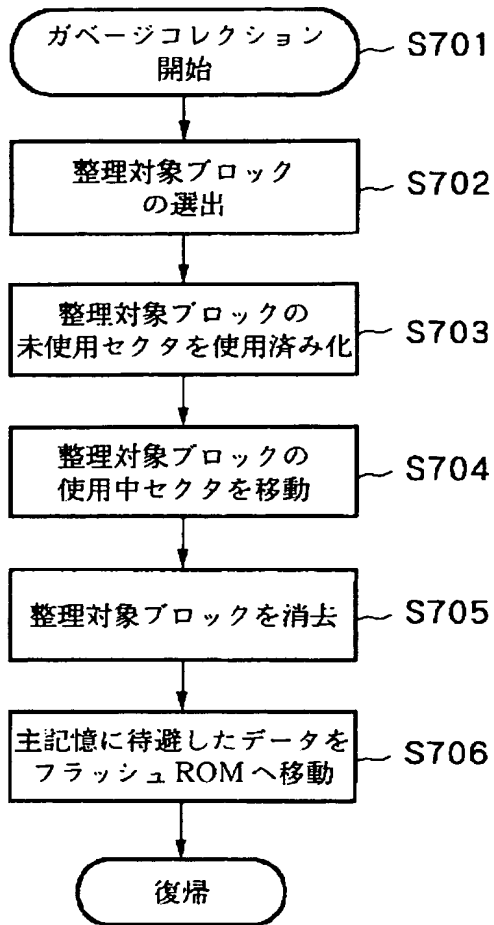
【図 3 4】



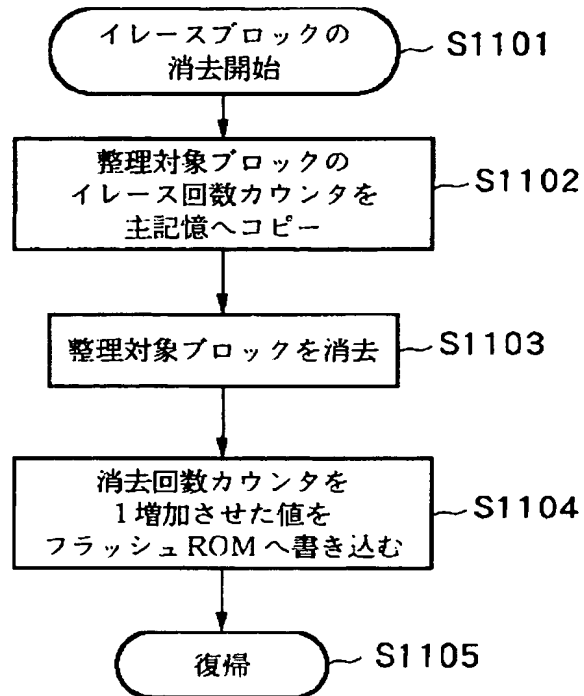
【図 4 6】



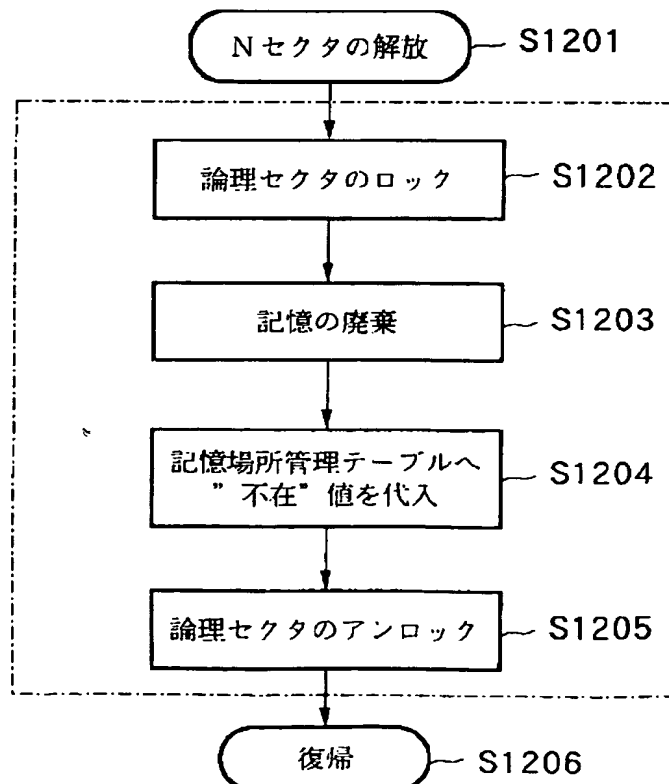
【図 2 4】



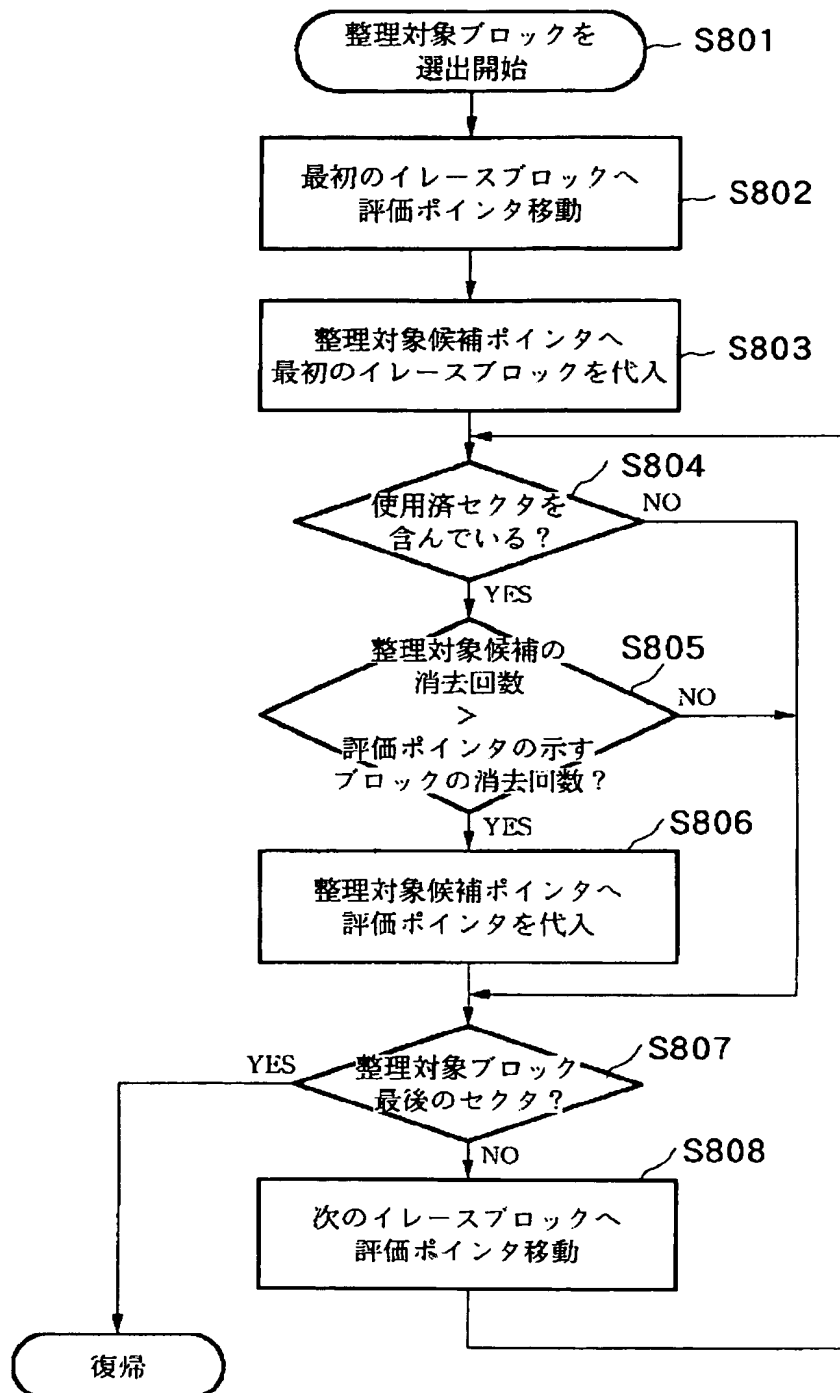
【図 2 8】



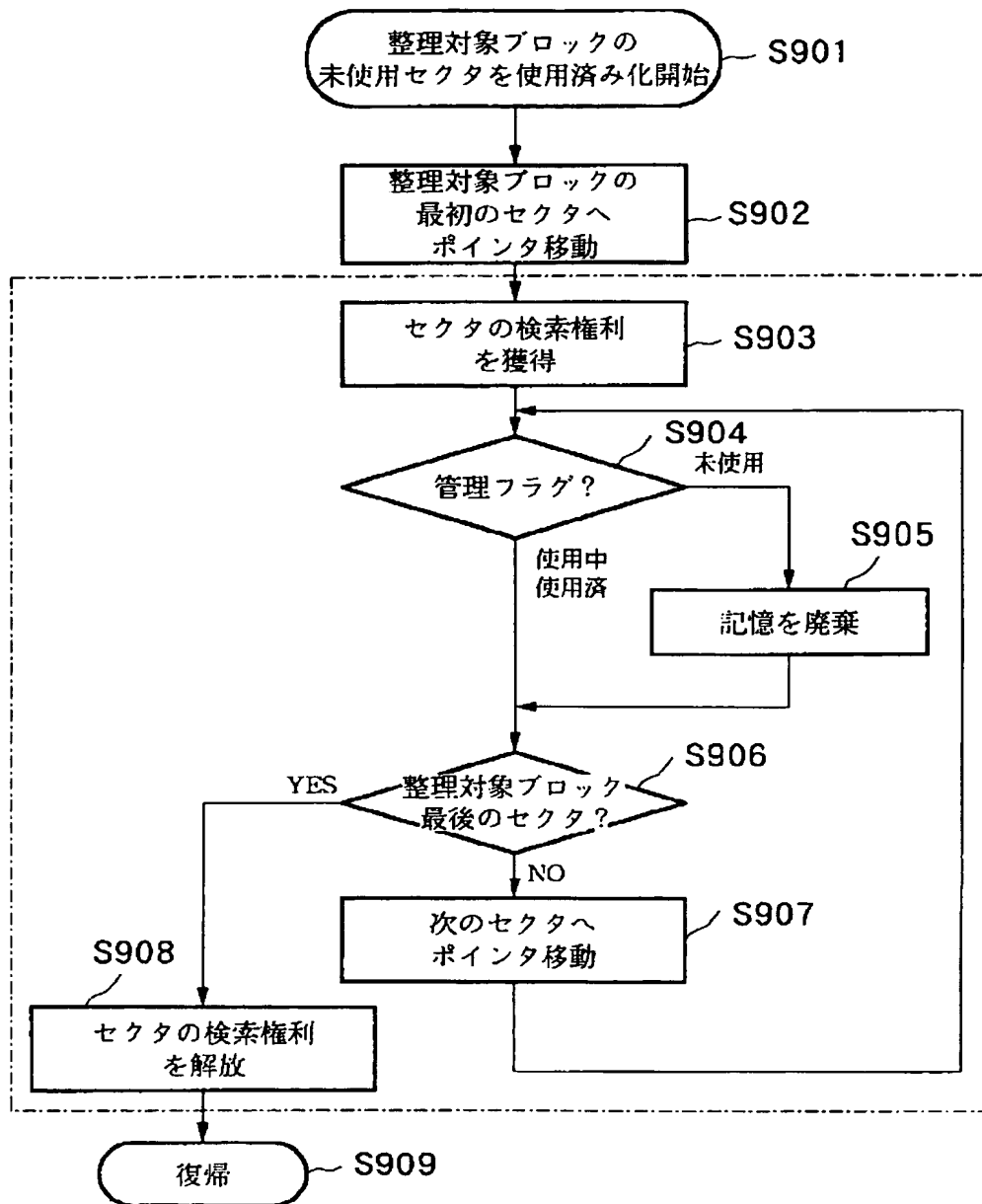
【図 2 9】



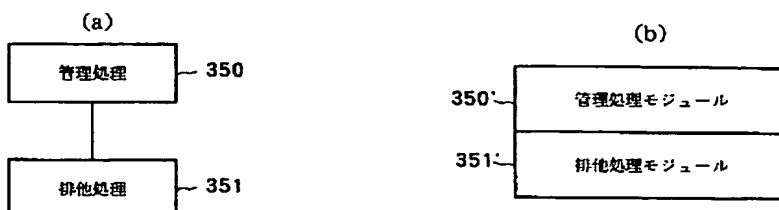
【図 2 5】



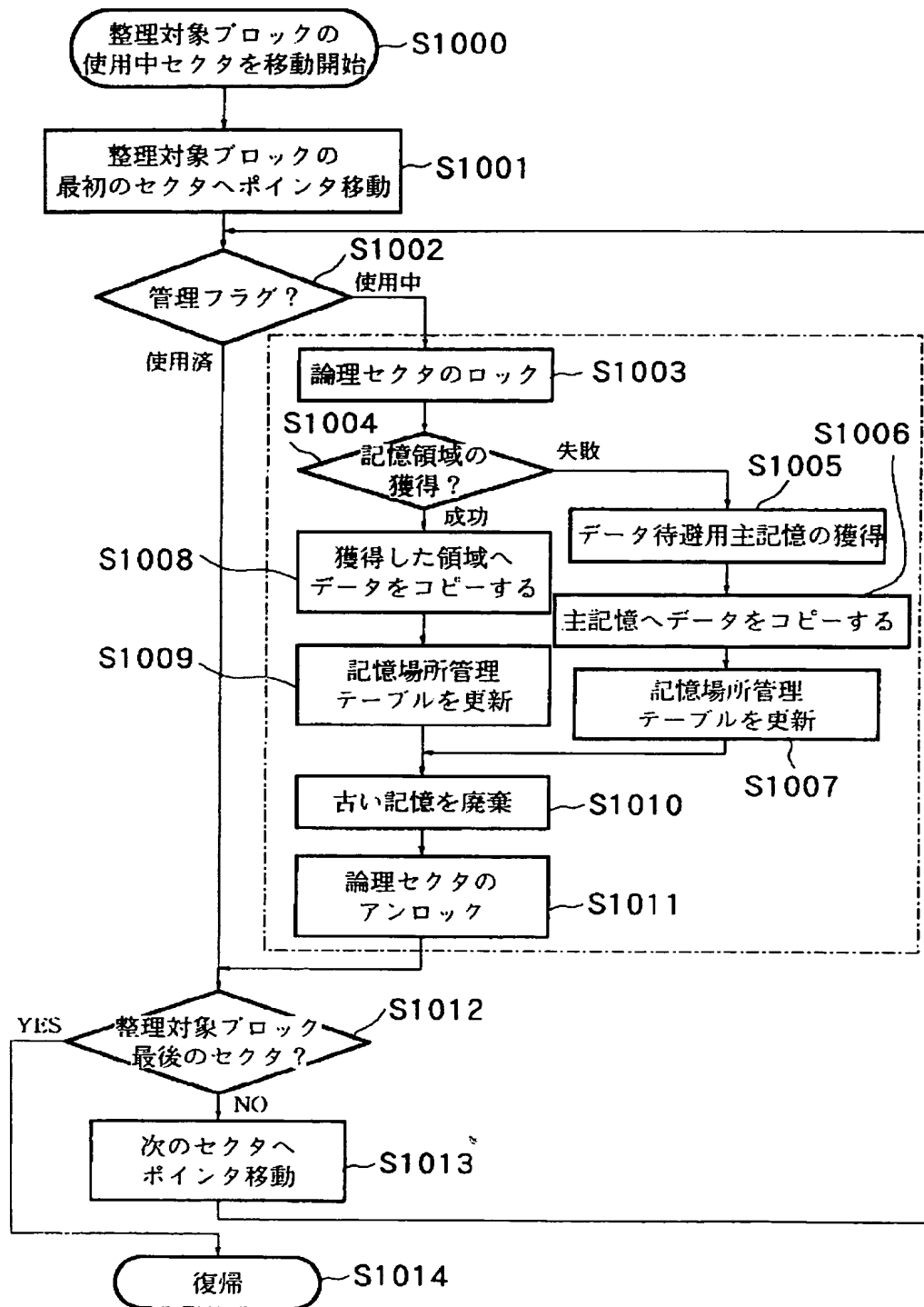
【図 2 6】



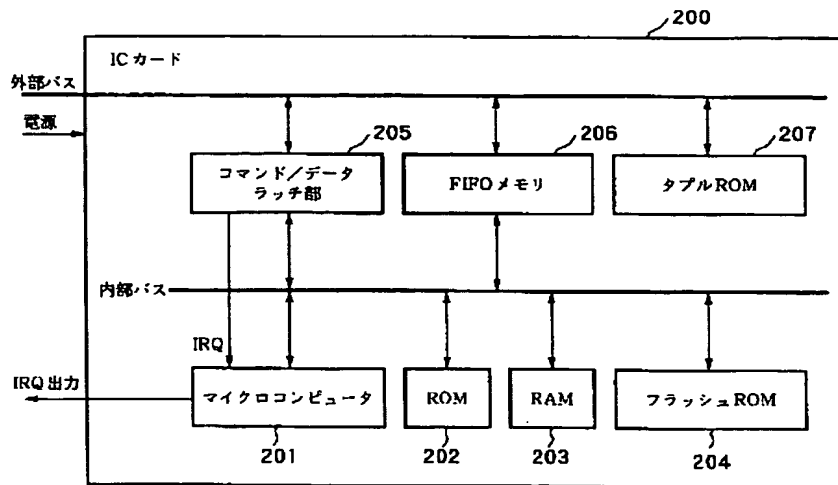
【図 4 9】



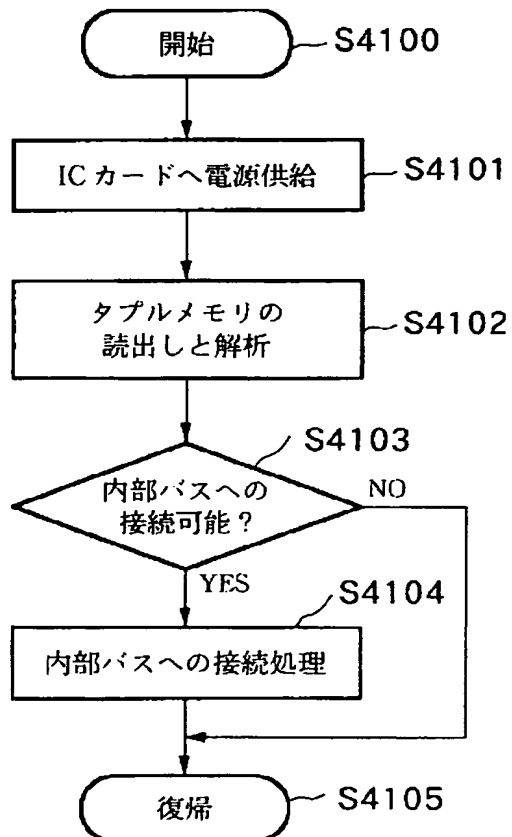
【図 2 7】



【図 3 0】



【図 3 2】



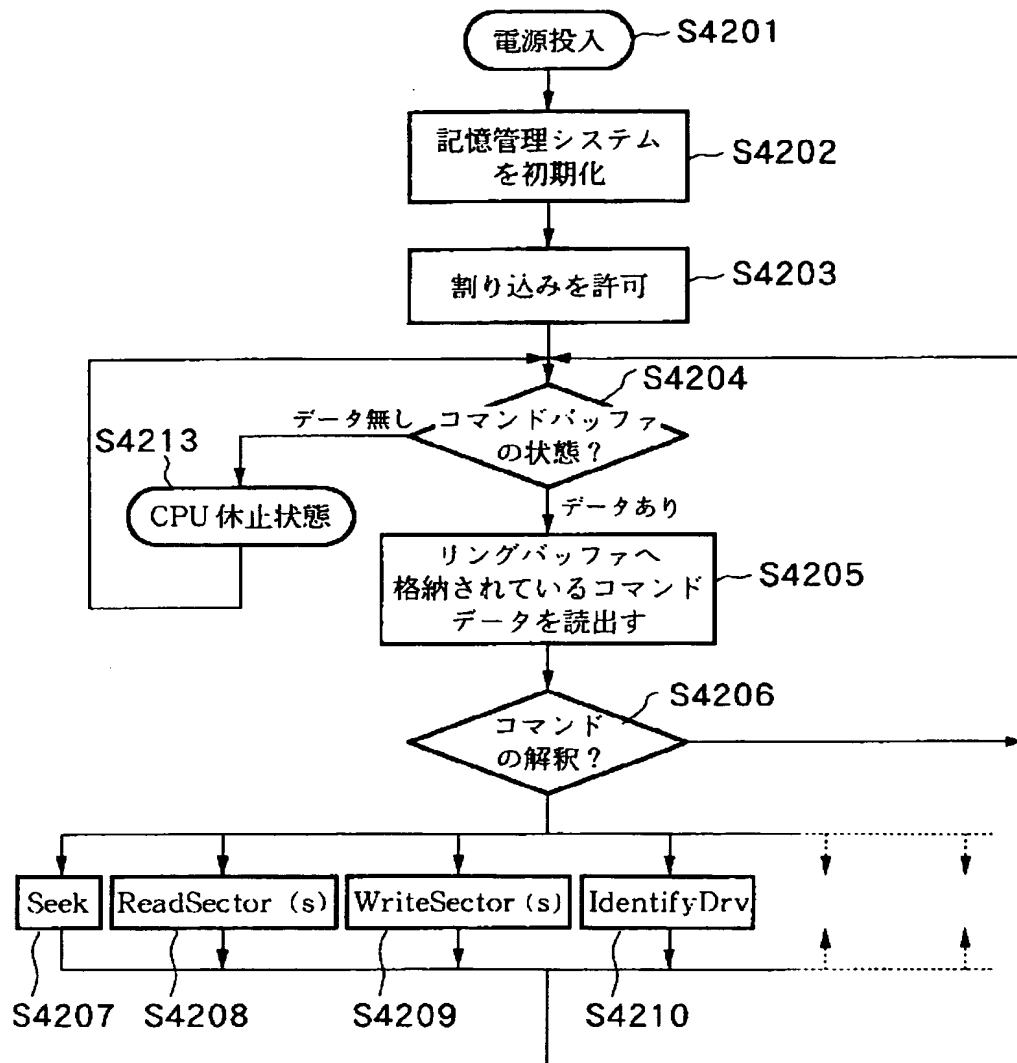
【図 3 5】

コマンド	
読出し (In)	書き込み (Out)
Error	Features
SectorCount	SectorCount
SectorNumber	SectorNumber
CylinderLow	CylinderLow
CylinderHigh	CylinderHigh
Drive/Head	Drive/Head
Status	Command

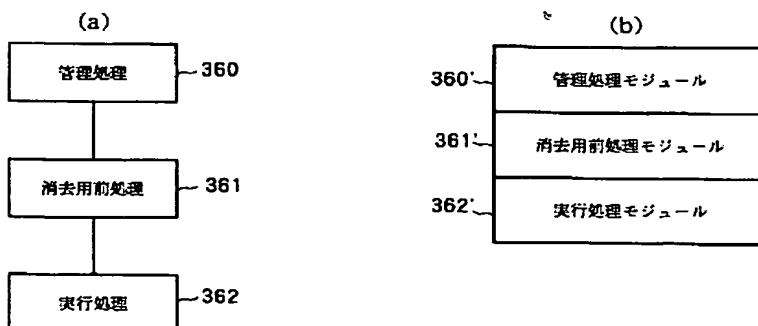
← 割り込み発生

コントロール	
読出し (In)	書き込み (Out)
Alt.Status	Device Ctl
DriveAddr	

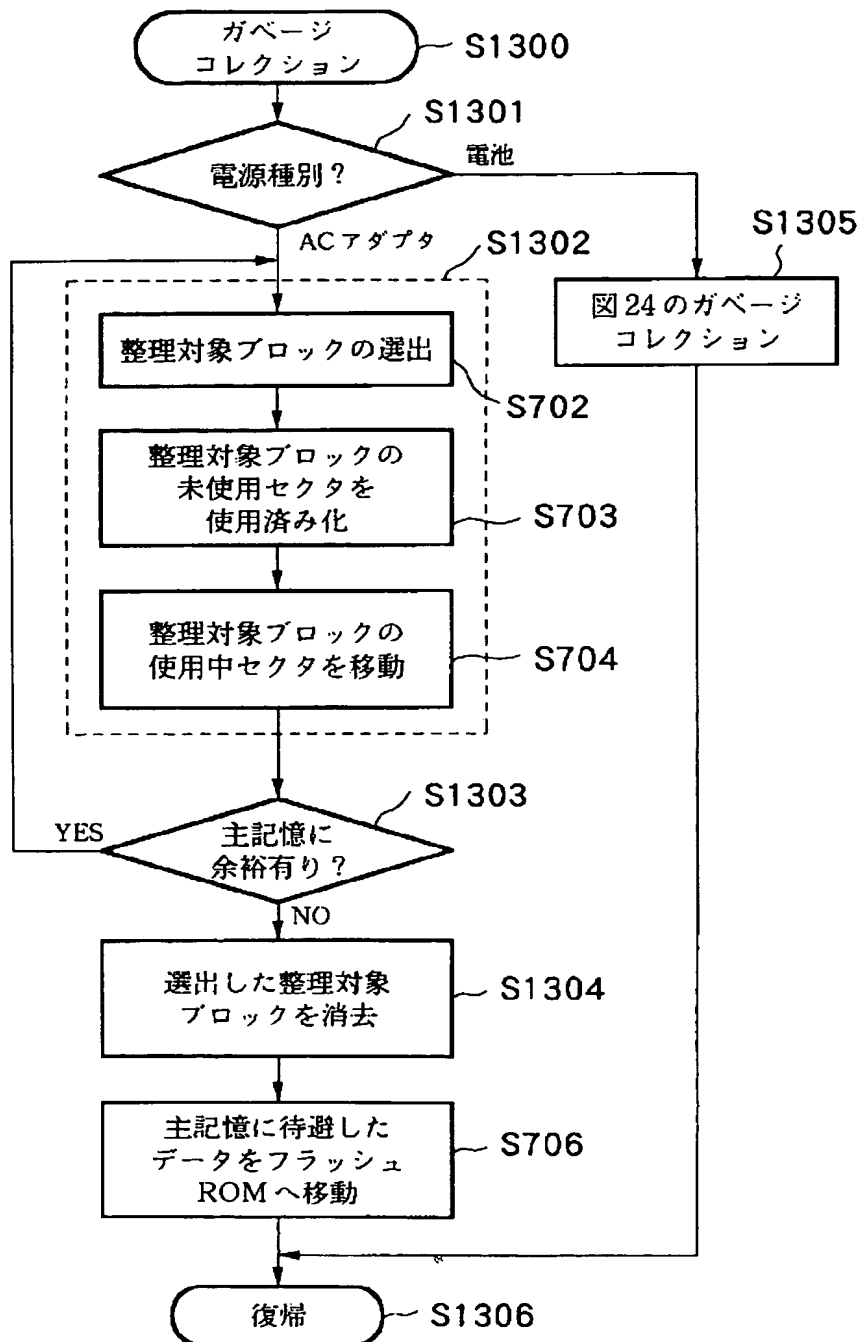
【図 3 3】



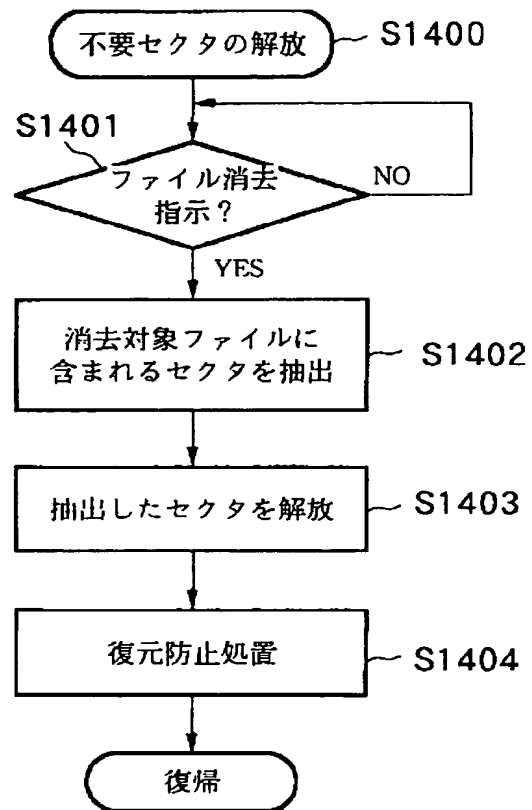
【図 5 0】



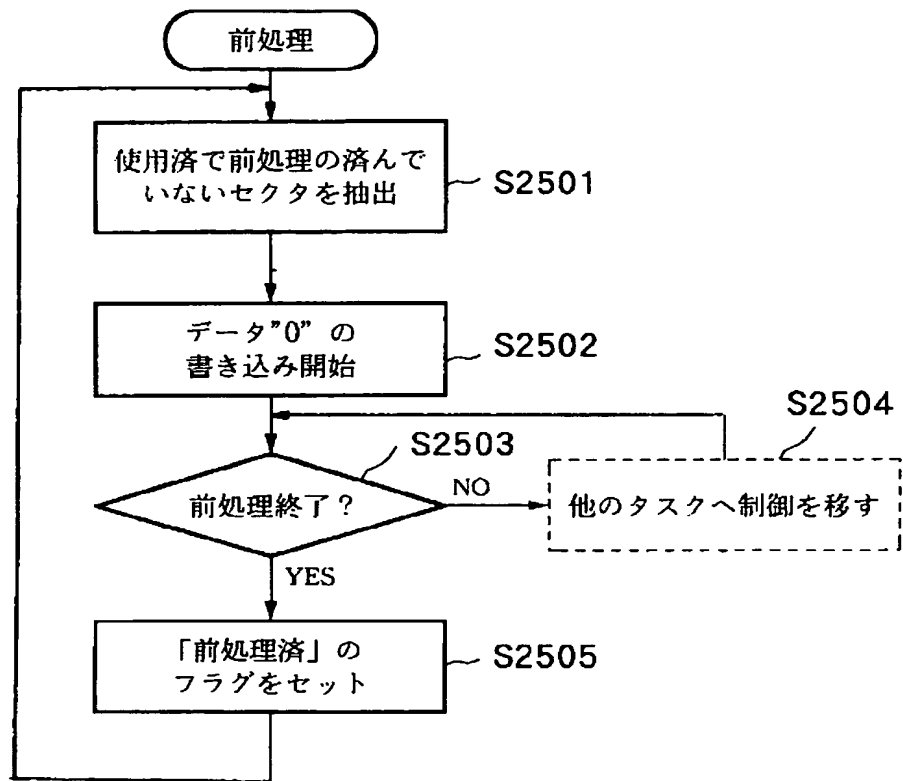
【図 3 6】



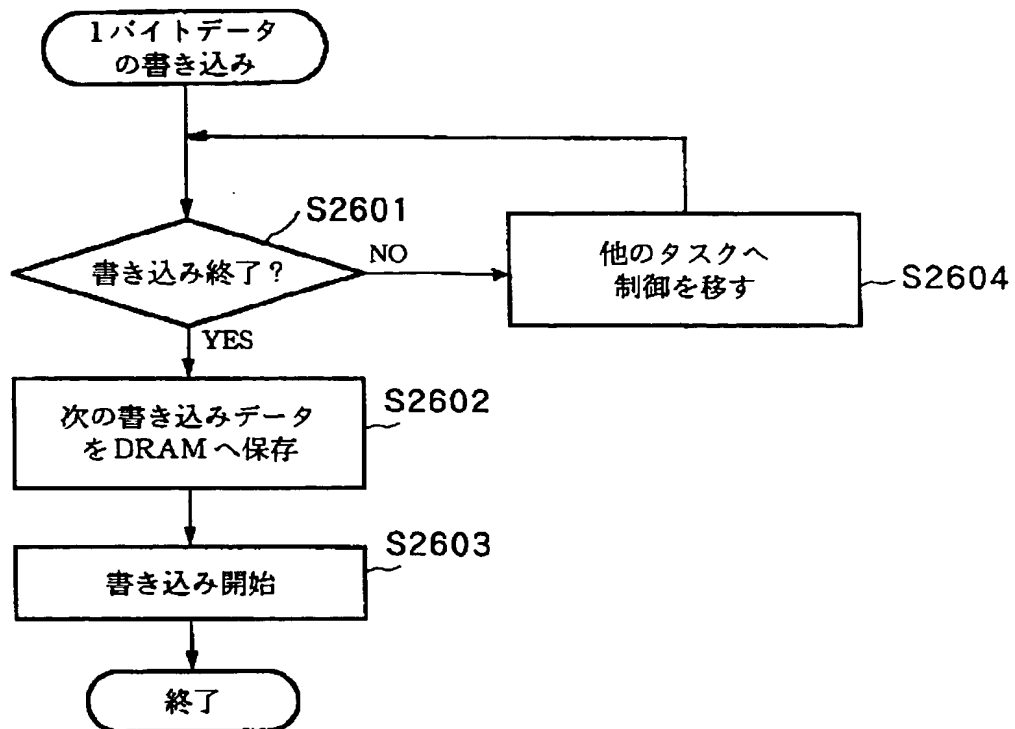
【図 3 7】



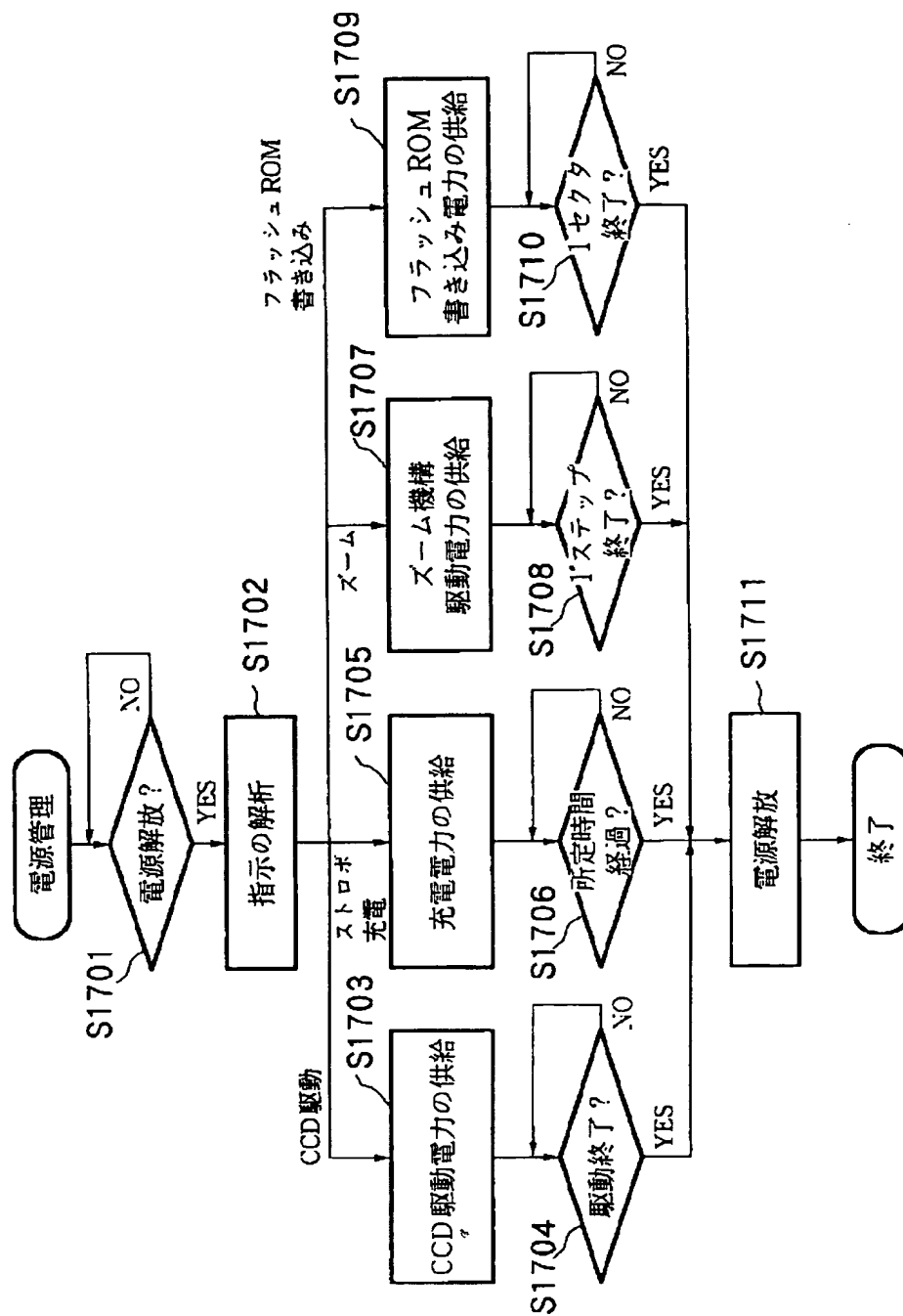
【図 3 8】



【図 3 9】



【図 40】



【図 48】

